Arie M. C. A. Koster
Xavier Muñoz (Eds.)

Graphs and Algorithms in Communication Networks

Studies in Broadband, Optical, Wireless and Ad Hoc Networks

Springer

cost

Texts in Theoretical Computer Science
An EATCS Series

Editors: W. Brauer   J. Hromkovič   G. Rozenberg   A. Salomaa

On behalf of the European Association
for Theoretical Computer Science (EATCS)

Advisory Board:
G. Ausiello   M. Broy   C.S. Calude   A. Condon
D. Harel   J. Hartmanis   T. Henzinger   T. Leighton
M. Nivat   C. Papadimitriou   D. Scott

Arie M.C.A. Koster · Xavier Muñoz

Editors

Graphs and Algorithms in Communication Networks

Studies in Broadband, Optical, Wireless
and Ad Hoc Networks

Springer

Editors

Prof. Dr. Ir. Arie M.C.A. Koster
Lehrstuhl II für Mathematik
RWTH Aachen
Germany
koster@math2.rwth-aachen.de

Prof. Xavier Muñoz
Dept. de Matemàtica Aplicada IV
Universitat Politècnica de Catalunya
Barcelona
Spain
xml@ma4.upc.edu

Cover design: KuenkelLopka GmbH

Printed on acid-free paper

COST

COST, the acronym for European COoperation in the field of Scientific and Technical Research, is the oldest and widest European intergovernmental network for cooperation in research. Established by the Ministerial Conference in November 1971, COST is presently used by the scientific communities of 35 European countries to cooperate in common research projects supported by national funds.

The funds provided by COST, less than 1% of the total value of the projects, support the COST cooperation networks (COST Actions) through which, with 30 million Euro per year, more than 30,000 European scientists are involved in research having a total value which exceeds two billion Euro per year. This is the financial worth of the European added value which COST achieves.

A "bottom-up approach" (the initiative of launching a COST Action comes from the European scientists themselves), "à la carte participation" (only countries interested in the Action participate), "equality of access" (participation is open also to the scientific communities of countries not belonging to the European Union) and "flexible structure" (easy implementation and light management of the research initiatives) are the main characteristics of COST.

As a precursor of advanced multidisciplinary research, COST has a very important role for the realization of the European Research Area (ERA), anticipating and complementing the activities of the Framework Programmes, constituting a "bridge" to the scientific communities of emerging countries, increasing the mobility of researchers across Europe, and fostering the establishment of "Networks of Excellence" in many key scientific domains such as Biomedicine and Molecular Biosciences; Food and Agriculture; Forestry Products and Services; Materials, Physical, and Nanosciences; Chemistry and Molecular Sciences and Technologies; Earth System Science and Environmental Management; Information and Communication Technologies; Transport and Urban Development; Individuals, Societies, Cultures, and Health. It covers basic and more applied research and also addresses issues of pre-normative nature or of societal importance.

Web: `www.cost.esf.org`

 ESF provides the COST office through an EC contract

 COST is supported by the EU RTD Framework programme

Preface

Communication networks are a vital and crucial element of today's world. Mobile devices, the Internet, and all new applications and services provided by these media have changed dramatically the way both individual lives and society as a whole are organized. All these services depend on fast and reliable data connections, whether wired or wireless. To meet such requirements, information and communication technology is challenged again and again to provide faster protocols, wireless interfaces with higher bandwidth capacity, innovative mechanisms to handle failures, and so on.

For many of those challenges a variety of mathematical disciplines contribute in a supportive role, either in providing insights, evidence, or algorithms or as decision support tools. In particular, the broad area of algorithmic discrete mathematics plays a crucial role in the design and operation of communication networks. However, the discipline is fragmented between scientific disciplines such as pure mathematics, theoretical computer science, distributed computing, and operations research. Furthermore, researchers from communication engineering utilize discrete mathematical techniques and develop their own extensions.

With the aim to bring together the above-mentioned disciplines and draw synergy effects from it, the COST action 293 – Graphs and Algorithms in Communication Networks – was launched in October 2004 for a period of four years. Scientists from the above disciplines have been gathering on a regular basis to learn from each other and to work jointly on emerging applications to the benefit of the information and communication technology society. Also workshops and training schools have been organized to disseminate recent advances in all subject areas. An active exchange programme (short-term scientific missions in COST terminology) between the research groups has resulted in a high number of joint publications.

To document on the one hand the multidisciplinary research carried out within COST 293 and on the other hand to encourage further collaborations between the disciplines, this book presents a number of studies in broadband, optical, wireless, and ad hoc networks where the techniques of algorithmic discrete mathematics have provided highly recognized contributions.

The way the studies are presented, this book is particularly suited for Ph.D. students, postdoctoral researchers in mathematics, computer science, operations research, and network engineering as well as industrial researchers who would like to investigate state-of-the-art mathematical alternatives to resolve the technological challenges of tomorrow. An introductory chapter should ease access to the material for researchers not familiar with the mathematical terminology used by the chapters' authors.

As chair and vice-chair of COST 293, it has been a pleasure for us to prepare this book. We would like to thank all authors and reviewers for the contributions. Without their voluntary help it would have been impossible to publish this book. We also are grateful to COST for supporting our action in general and the dissemination of this book in particular .

Coventry/Barcelona,   *Arie M.C.A. Koster*
March 2009   *Xavier Muñoz*

Contents

Part I Studies in Broadband and Optical Networks

8 Optimization of OSPF Routing in IP Networks 199

Andreas Bley, Bernard Fortz, Eric Gourdin, Kaj Holmberg, Olivier
Klopfenstein, Michał Pióro, Artur Tomaszewski, and Hakan Ümit

9 Game-Theoretic Approaches to Optimization Problems in Communication Networks 241

Vittorio Bilò, Ioannis Caragiannis, Angelo Fanelli, Michele Flammini,
Christos Kaklamanis, Gianpiero Monaco, and Luca Moscardelli

10   Permutation Routing and (ℓ,k)-Routing on Plane Grids 265
Ignasi Sau and Janez Žerovnik

Part II Studies in Wireless and Ad Hoc Networks

11   Mathematical Optimization Models for WLAN Planning 283
Sandro Bosio, Andreas Eisenblätter, Hans-Florian Geerdes, Iana
Siomina, and Di Yuan

List of Contributors

Georg Baier
Siemens AG, Munich, Germany, e-mail: `georg.baier@siemens.com`

Pietro Belotti
Dept. of Industrial & Systems Engineering, Lehigh University, Bethlehem, PA, USA, e-mail: `belotti@lehigh.edu`

Vittorio Bilò
Department of Mathematics, University of Salento, Provinciale Lecce-Arnesano, P.O. Box 193, 73100 Lecce, Italy, e-mail: `vittorio.bilo@unile.it`

Andreas Bley
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany.
Current address: Institute for Mathematics, Technical University Berlin, Str. des 17. Juni 136, D-10623 Berlin, Germany, e-mail: `bley@math.tu-berlin.de`

Vincenzo Bonifaci
Dipartimento di Ingegneria Elettrica e dell'Informazione, Università degli Studi dell'Aquila, Poggio di Roio, 67040 L'Aquila, Italy
Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma, Italy, e-mail: `bonifaci@dis.uniroma1.it`

Sandro Bosio
Institut für Mathematische Optimierung, Otto-von-Guericke Universität, D-39106 Magdeburg, Germany, e-mail: `bosio@mail.math.uni-magdeburg.de`

Eusebi Calle
Institute of Informatics and Applications, University of Girona, Campus Montilivi, 17071 Girona, Spain, e-mail: `eusebi@eia.udg.es`

Ioannis Caragiannis
Computer Technology Institute & Department of Computer Engineering and Informatics, University of Patras, 26500 Rio, Greece, e-mail: `caragian@ceid.upatras.gr`

Luis Fernando Caro
Institute of Informatics and Applications, University of Girona, Campus Montilivi,
17071 Girona, Spain, e-mail: `lfcaro@atc.udg.edu`

Lenka Carr-Motyckova
Department of Computer Science and Electrical Engineering, Luleå University of
Technology, SE-971 87 Luleå, Sweden, e-mail: `lenka@sm.luth.se`

Tibor Cinkler
Department of Telecommunications and Media Informatics, Bu-
dapest University of Technology and Economics, Budapest, Hungary,
e-mail: `cinkler@tmit.bme.hu`

David Coudert
MASCOTTE, INRIA, I3S, CNRS UMR6070, University of Nice-Sophia Antipolis,
France, e-mail: `David.Coudert@sophia.inria.fr`

Yezid Donoso
Computer Science and Engineering Department, University of Los Andes, Bogotá,
Colombia, e-mail: `ydonoso@uninorte.edu.co`

Andreas Eisenblätter
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany, e-mail:
`eisenblaetter@zib.de`, and
atesio GmbH, Sophie-Taeuber-Arp-Weg 27, D-12205 Berlin, Germany, e-mail:
`eisenblaetter@atesio.de`

Thomas Engel
Nokia Siemens Networks GmbH & Co. KG, Munich, Germany, e-mail:
`thomas.1.engel@nsn.com`

Ramón Fabregat
Institute of Informatics and Applications, University of Girona, Campus Montilivi,
17071 Girona, Spain, e-mail: `ramon@silver.udg.es`

Angelo Fanelli
Dipartimento di Informatica, Università degli Studi dell'Aquila, Via Vetoio,
Loc. Coppito, 67100 L'Aquila, Italy,
e-mail: `angelo.fanelli@di.univaq.it`

Michele Flammini
Dipartimento di Informatica, Università degli Studi dell'Aquila, Via Vetoio,
Loc. Coppito, 67100 L'Aquila, Italy, e-mail: `flammini@di.univaq.it`

Bernard Fortz
Départment d'Informatique, Faculté des Sciences, Université Libre de Bruxelles
(ULB), Belgium, and
CORE, Université catholique de Louvain, Belgium,
e-mail: `bernard.fortz@ulb.ac.be`

Jérôme Galtier
Orange Labs, 905 Avenue Albert Einstein, 06921 Sophia Antipolis Cedex, France,
e-mail: `jerome.galtier@orange-ftgroup.com`

Hans-Florian Geerdes
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany

Eric Gourdin
France Telecom, Orange Labs R&D, France,
e-mail: `eric.gourdin@orange-ftgroup.com`

Xavier Hesselbach
Deparment of Telematics Engineering, Universitat Politècnica de Catalunya,
C/Jordi Girona, 1 i 3, Mòdul C3 – Campus Nord, 08034 Barcelona, Spain, e-mail:
`xavierh@entel.upc.edu`

Kaj Holmberg
Linköping Institute of Technology, SE-581 83 Linköping, Sweden, e-mail:
`kahol@mai.liu.se`

Mónica Huerta
Electronics and Circuits Department, University Simón Bolívar, Caracas,
Venezuela, e-mail: `mhuerta@usb.ve`

Tomas Johansson
Department of Computer Science and Electrical Engineering, Luleå University of
Technology, SE-971 87 Luleå, Sweden, e-mail: `tomasjo@sm.luth.se`

Alpár Jüttner
Department of Operations Research, Eötvös University, Pázmány P. s. 1/C, H-1117
Budapest, Hungary, e-mail: `alpar@cs.elte.hu`

Christos Kaklamanis
Computer Technology Institute & Department of Computer Engineer-
ing and Informatics, University of Patras, 26500 Rio, Greece, e-mail:
`kakl@ceid.upatras.gr`

Ralf Klasing
CNRS - LaBRI - Université Bordeaux 1, France, e-mail: `klasing@labri.fr`

Mirosław Klinkowski
Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya,
Barcelona, Spain, and
Department of Transmission and Optical Technology, National Institute of
Telecommunications, Warsaw, Poland, e-mail: `mklinkow@ac.upc.edu`

Olivier Klopfenstein
France Telecom, Orange Labs R&D, France,
e-mail: `olivier.klopfenstein@orange-ftgroup.com`

Christos Kolias
Covad Communications, 110 Rio Robles, San Jose, CA 95134, USA, e-mail:
`ckolias@gmail.com`

Peter Korteweg
Department of Mathematics and Computer Science, Eindhoven University of
Technology, The Netherlands, e-mail: `peterkorteweg@hotmail.com`

Arie M. C. A. Koster
Centre for Discrete Mathematics and its Applications (DIMAP), Warwick Business
School, University of Warwick, Coventry CV4 7AL, United Kingdom.
Current address: Lehrstuhl II für Mathematik, RWTH Aachen Univer-
sity, Wüllnerstr. zwischen 5 und 7, D-52062 Aachen, Germany, e-mail:
`koster@math2.rwth-aachen.de`

Alberto Marchetti-Spaccamela
Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma, Italy,
e-mail: `alberto@dis.uniroma1.it`

Marian Marciniak
Department of Transmission and Optical Technology, National Institute of
Telecommunications, Warsaw, Poland, e-mail: `m.marciniak@itl.waw.pl`

José L. Marzo
Institute of Informatics and Applications, University of Girona, Campus Montilivi,
17071 Girona, Spain, e-mail: `joseluis.marzo@udg.edu`

Gianpiero Monaco
Dipartimento di Informatica, Università degli Studi dell'Aquila, Via Vetoio,
Loc. Coppito, 67100 L'Aquila, Italy,
e-mail: `gianpiero.monaco@di.univaq.it`

Luca Moscardelli
Dipartimento di Informatica, Università degli Studi dell'Aquila, Via Vetoio,
Loc. Coppito, 67100 L'Aquila, Italy, e-mail: `moscardelli@di.univaq.it`

Xavier Muñoz
Graph Theory and Combinatorics Group, Department of Applied Mathe-
matics IV, Universitat Politècnica de Catalunya, Barcelona, Spain, e-mail:
`xml@ma4.upc.edu`

Alfredo Navarra
Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Via
Vanvitelli 1, 06123 Perugia, Italy, e-mail: `navarra@dmi.unipg.it`

Sebastian Orlowski
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany.
Current address: atesio GmbH, Sophie-Taeuber-Arp-Weg 27, D-12205 Berlin,
Germany, e-mail: `orlowski@atesio.de`

Dimitri Papadimitriou
Network and Technology Strategy, Alcatel-Lucent, Copernicuslaan 50, B-2018
Antwerpen, Belgium,
e-mail: `dimitri.papadimitriou@alcatel-lucent.be`

David Peleg
Department of Computer Science and Applied Mathematics, The Weizmann Insti-
tute of Science, Rehovot, Israel, e-mail: `david.peleg@weizmann.ac.il`

Michał Pióro
Institute of Telecommunications, Warsaw University of Technology, ul.
Nowowiejska 15/19, 00-665 Warsaw, Poland, and
Department of Electrical and Information Technology, Lund University, Box 118,
SE-221 00 Lund, Sweden, e-mail: `mpp@tele.pw.edu.pl`

Christian Raack
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany, e-mail:
`raack@zib.de`

Tomasz Radzik
Department of Computer Science, King's College London, London WC2R 2LS,
United Kingdom, e-mail: `tomasz.radzik@kcl.ac.uk`

Sarah Ruepp
Networks Competence Area, DTU Fotonik, Technical University of Denmark,
2800 Kgs. Lyngby, Denmark, e-mail: `sr@com.dtu.dk`

Ignasi Sau
Mascotte, INRIA, I3S, CNRS UMR6070, University of Nice-Sophia Antipolis,
France, and
Graph Theory and Combinatorics Group, Department of Applied Mathe-
matics IV, Universitat Politècnica de Catalunya, Barcelona, Spain, e-mail:
`Ignasi.Sau@sophia.inria.fr`

Mordechai Shalom
Tel-Hai Academic College, Upper Galilee, 12210, Israel, e-mail:
`cmshalom@cs.technion.ac.il`

Iana Siomina
Ericsson Research, Ericsson AB, Isafjordsgatan 14E, SE-164 80, Stockholm,
Sweden, e-mail: `iana.siomina@ericsson.com`

Fernando Solano
Institute of Telecommunications, Warsaw University of Technology, ul.
Nowowiejska 15/19, 00-665 Warsaw, Poland, e-mail: `fs@tele.pw.edu.pl`

Thomas Stidsen
Informatics and Mathematical Modeling, Danish Technical University, Richard
Petersens Plads, DK-2800 Kgs. Lyngby, Denmark, e-mail: `tks@imm.dtu.dk`

Leen Stougie
Department of Economics and Business Administration, Free University, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands, e-mail: lstougie@feweb.vu.nl, and
CWI, Amsterdam, The Netherlands, e-mail: stougie@cwi.nl

Janos Tapolcai
Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary, e-mail: tapolcai@tmit.bme.hu

Artur Tomaszewski
Institute of Telecommunications, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland, e-mail: artur@tele.pw.edu.pl

Hakan Ümit
Louvain Management School, Belgium, and
CORE, Université catholique de Louvain, Belgium,
e-mail: hakan.umit@uclouvain.be

Walter Unger
Lehrstuhl für Informatik 1, RWTH Aachen University, Ahornstraße 55, D-52056 Aachen, Germany, e-mail: quax@cs.rwth-aachen.de

Di Yuan
Department of Science and Technology, Linköping University, SE-601 74, Norrköping, Sweden, e-mail: diyua@itn.liu.se

Shmuel Zaks
Computer Science Department, Technion, Haifa, Israel, e-mail: zaks@cs.technion.ac.il

Janez Žerovnik
University of Maribor, Smetanova 17, Maribor 2000, Slovenia, and
Institute of Mathematics, Physics and Mechanics, Jadranska 19, Ljubljana, Slovenia, e-mail: janez.zerovnik@imfm.uni-lj.si

Acronyms

3-MECA	Maximum Efficiency Channel Assignment with Three Channels, Hyperbolic Formulation
3-MECA$_E$	Maximum Efficiency Channel Assignment with Three Channels, Linear Formulation
3-MOCA	Minimum Overlap Channel Assignment with three channels
3-MT-MO	Integration of MTAL and 3-MOCA
3-MT-ME	Integration of MTAL and 3-MECA
ABC	Adaptive Broadcast Consumption
ACK	Acknowledgement Frame
ADM	Add/Drop Multiplexer
AODV	Ad Hoc On Demand Distance Vector
AOLS	All-Optical Label Switching
AP	Access Point
API	Average Path Interference
APX	Approximable
AR	Alternative Routing
AS	Autonomous System
ATM	Asynchronous Transfer Mode
B&C	Branch-and-Cut
BEB	Binary Exponential Backoff
BIP	Broadcast Incremental Power
BL	Basic Localization
BLP	Burst Loss Probability
BSS	Basic Service Set
CBWFQ	Class-Based Weighted Fair Queueing
CDMA	Code Division Multiple Access
CFS	Cost Function Smoothing
CMAX	Capacity-Competitive Algorithm
CMI	Cost Minimization in Multi-interface Networks
CoS	Class of Service
CR-LDP	Constraint-Based Routing Label Distribution Protocol

CRP	Contention Resolution Protocol
CS	Circuit Switching
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CSPF	Constraint Shortest Path First
CTS	Clear to Send
CWDM	Coarse Wavelength Division Multiplexing
DAG	Directed Acyclic Graph
DCF	Distributed Coordination Function
DiffServ	Differentiated Services
DIFS	Distributed Inter-frame Space
DPP	Dedicated Path Protection
DS	Distribution System
DWDM	Dense Wavelength Division Multiplexing
D-LSP	Distributed LSP
ECMP	Equal Cost Multi-path
ECS	Effective Computing System
ELS	Ethernet VLAN-Label Switching
ESS	Extended Service Set
EXC	Electrical Cross-connect
FAP	Frequency Assignment Problem
FDM	Frequency Division Multiplexed
FDMA	Frequency Division Multiple Access
FDPP	Failure Dependant Path Protection
FEC	Forwarding Equivalence Class
FIFO	First-In First-Out
FIP	Finite Improvement Path
FPQ	Fair Packet Queueing
FSC	Fiber Switching Capable
GbE	Gigabit Ethernet
Gbit/s	Gigabit per Second
GHz	Gigahertz
GMM	Generalized Multicast Multi-path
GMPLS	Generalized MPLS
GNPP	General Network Planning Problem
GSM	General System for Mobile Communication
IBM	Induced Bipartite Matching
IBSS	Independent Basic Service Set
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
ILP	Integer Linear Programming
IMBM	Iterative Maximum-Branch Minimization
IP	Internet Protocol
IPv4	Internet Protocol Version 4

IS-IS	Intermediate System to Intermediate System
ISP	Inverse Shortest Path Problem
ITU	International Telecommunication Union
JET	Just-Enough-Time
JIT	Just-In-Time
LDP	Label Distribution Protocol
LER	Label Edge Router
L2SC	Layer 2 Switching Capable
LIB	Label Information Base
LISE	Low Interference Spanner Establisher
LL	(Overall) Link Loss
LL-NRL	Link Loss model with Non-Reduced Load
LP	Linear program
LSP	Label Switched Path
LSR	Label Switched Router
LTE	Light Termination Equipment
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
Mbit/s	Megabit per second
MCNFP	Multi-commodity Network Flow Problem
MEBR	Minimum Energy Broadcast Routing
MECA	Maximum Efficiency Channel Assignment, hyperbolic formulation
MERLIN	Mergin Link group
MILP	Mixed-Integer Linear Program
MIP	Mixed-Integer Program
MIR	Mixed-Integer Rounding
MIRA	Minimum Interference Routing Algorithm
MLTE	Multilayer Traffic Engineering
MLU	Maximum Link Utilization
MOCA	Minimum Overlap Channel Assignment
MOP	Multi-Objective Problem
MPLS	Multi-protocol Label Switching
MPLS-TE	Multi-protocol Label Switching Traffic Engineering
MR	Multi-path Routing
MST	Minimum Spanning Tree
MSTP	Minimum Spanning Tree Protocol
MT	Mobile Terminal
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
MT-MO	Integration of MTAL and MOCA
NHLFE	Next Hop Label Forwarding Entry
NL	(Overall) Network Loss
NL-RL	Network Loss model with Reduced Load
NL-NRL	Network Loss model with Non-Reduced Load
NP	Non-deterministic Polynomial

NRL	Non-Reduced Load
NSF	National Science Foundation
OBS	Optical Burst Switching
OFDM	Orthogonal Frequency Division Multiplexing
OSPF	Open Shortest Path First protocol
OTIS	Optical Transpose Interconnection System
OTN	Optical Transport Network
oVPN	Open Virtual Private Network
OXC	Optical Cross-Connect
P2P	Peer-to-peer
PCF	Point Coordination Function
PIRA	Path-Interfering Routing Algorithm
PSC	Packet Switching Capable
PTAS	Polynomial Time Approximation Scheme
QoS	Quality of Service
RCFS	Randomized Cost Function Smoothing
RFC	Request for Comment
RIT	Reservation with Just-In-Time
RL	Reduced Load
RSVP	Resource Reservation Protocol
RSVP-TE	Resource Reservation Protocol for Traffic Engineering
RTS	Request to Send
SAT	Satisfiability problem
SCFQ	Self-Clocked Fair Queueing
SCSP	Shortcut Span Protection
SDH	Synchronous Digital Hierarchy
SFQ	Start-time Fair Queueing
SIFS	Short Inter-Frame Space
SLP	Shared Link Protection
SONET	Synchronous Optical Network
SPP	Shared Path Protection
SPR	Shortest Path Routing
SPT	Shortest Path Tree
SRG	Shared Risk Group
SSP	Shared Segment Protection
STEP	Shortest Path Traffic Engineering Problem
STM	Synchronous Transport Module
TAG	Tell-And-Go
TAW	Tell-And-Wait
TCP	Transmission Control Protocol
TDM	Time Division Multipling
TDMA	Time Division Multiple Access
TE	Traffic Engineering
ToA	Time of Arrival
TP	Test Point

TSC	TDM Switching Capable
UDG	Unit Disk Graph
UMTS	Universal Mobile Telecommunications System
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
VPλN	Virtual Private Wavelength Network
VON	Virtual Overlay Network
WBSC	WaveBand Switching Capable
WDM	Wavelength Division Multiplexing
WFQ	Weighted Fair Queueing
WGP	Wireless Gathering Problem
WLAN	Wireless Local Access Network
WλSC	Wavelength Switching Capable
XTC	X Topology Control

Part I
Studies in Broadband and Optical Networks

Chapter 1
Graphs and Algorithms in Communication Networks on Seven League Boots

Arie M. C. A. Koster and Xavier Muñoz

Abstract This chapter provides an introduction to the mathematical techniques used to provide insight and decision support in the design and operaton of communication networks. Techniques discussed include graph-theoretical concepts, (integer) linear programming, and complexity theory. To illustrate the importance of these techniques, classical applications in the area of communication networks are discussed. The wide variety and depth of the mathematics involved does not allow an exposition highlighting all details. References for further reading are provided. The chapter is closed with a brief description of the applications discussed in the consecutive chapters.

Key words: combinatorial optimization, graph theory, networks, topology design, routing, network planning, frequency assignment, network coverage

1.1 Introduction

Graphs and algorithms play a vital role in modern communication networks. Without the mathematical theory and algorithms developed by researchers from discrete mathematics, algorithmics, mathematical optimization, and distributed computing, many services of the information society like (mobile) telephony, virtual private networks, broadband at home, wireless Internet access, and Phone over IP are unthinkable in their current form. At the heart of each of these, graphs are used to specify

Arie M. C. A. Koster

Lehrstuhl II für Mathematik, RWTH Aachen University, Wüllnerstr. zwischen 5 und 7, D-52062 Aachen, Germany, e-mail: `koster@math2.rwth-aachen.de`

Xavier Muñoz

Graph Theory and Combinatorics Group, Department of Applied Mathematics IV, Universitat Politècnica de Catalunya, Barcelona, Spain, e-mail: `xml@ma4.upc.edu`

the networks and technological features, whereas algorithms are used to compute solutions for cost-efficient design and smooth operation of the technologies.

The field of *discrete mathematics* deals with discrete structures such as graphs, hyper-graphs, and general combinatorial designs (e.g., balanced incomplete block designs, group divisible designs, etc.), which represent excellent instruments for modeling complex processes such as communication networks in an abstract, concise, and precise manner, concentrating on their relevant properties in order to analyze situations and elaborate problem solutions. While physical or virtual networks naturally allow for modeling with graphs, graphs are also used to describe more abstract relations such as the conflict between the various elements of a communication network (e.g., interference between antennae; see Section 1.5.5.1). Parameters defined for these discrete structures characterize not only the structures, but also furnish essential information on the applications being investigated. Moreover, powerful tools can be developed to solve practical problems by adapting core algorithms stemming from the discrete mathematics field.

An *algorithm* is a procedural step-by-step description to answer questions that are too complex to be solved instantly. When a numerical answer is expected, the algorithmic steps typically involve elementary computations. As the complexity of the question increases, the need for algorithms that require as few elementary computations as possible increases as well. Although modern computers allow for millions of computations in a short time frame, certain algorithms are still too time consuming to answer practical relevant questions.

Motivated by practical problems to be solved, the study of *efficient algorithms* is one of the most prolific and successful fields of computer science. Besides efficient algorithms, it has generated several important concepts, such as the notions of randomized algorithm, NP-hard optimization problem, and approximation algorithm. Typically, the field explores the design of an efficient (deterministic or randomized) algorithm for the problem at hand. If the problem is NP-hard, it resorts to the development of an approximation algorithm (see Section 1.3 for further details). The study of online versions of these algorithms has become an important stream of research, since the problem input is not known in advance in most applications.

The field of *mathematical optimization* deals with the development and implementation of optimization algorithms to support (quantitative) decisions. In communication networks, mathematical optimization is primarily applied to network design problems in wireless and broadband networks. Typical tasks for which mathematical optimization assists decision makers are the cost-effective design of network infrastructures, the reduction of interference in wireless networks, the area-wide introduction of digital broadcasting, and the determination of routing weights in OSPF Internet routing. Mathematical optimization (as well as other fundamental areas) also may help in identifying bottlenecks in systems and in conceiving workarounds and suggesting possible improvements. Optimization is also important in terms of economics and other business aspects related to communication networks.

Many (network) optimization problems can be modeled by means of a graph, and the decisions have a discrete structure. In such cases, a combinatorial optimization problem has to be solved. One branch of mathematical optimization focuses on the

study of the polyhedral structure of such problems. The more that is known about the structure of the problem, the more efficiently the problem can be solved.

The field of *distributed computing* is devoted to the structural and algorithmic problems arising from the exploitation of distributed systems of computers by means of communication networks. The range of applicability of this area of research spans from the cluster-computing paradigm that deals with a small number of computers, possibly within a company and connected by a high-bandwidth LAN, to the peer-to-peer (P2P) paradigm, through which millions of computers may be connected over the Internet. Topics under study range from basic research on impossibility results for asynchronous systems, to the most recent advances on the survivability of P2P networks.

All these areas are closely related. To name a few relations, mathematical optimization often exploits graph structures, algorithms are studied for distributed computing systems, mathematical optimization algorithms are analyzed on their strengths and weaknesses, and the scalability of distributed systems can be improved using tools from graph theory. In this chapter we would like to give a brief introduction to each of these tools from the mathematical toolbox. In Section 1.2, the framework of mathematical modeling by graphs and networks, including combinatorial and nonlinear optimization models and distributed and online problems, is introduced. Next, the complexity of algorithms is discussed in Section 1.3 before the most common methodologies to solve (combinatorial) optimization problems are presented in Section 1.4. To illustrate the use of these techniques, Section 1.5 provides a range of classical applications of graphs and algorithms in communication networks. Where appropriate, references are made to more advanced applications in forthcoming chapters.

1.2 Mathematical Modeling

In this section we introduce the most common mathematical structures used to model communication networks and their decision problems. First, concepts such as (un)directed graphs and networks are introduced, and their substructures are defined. Next, the concept of a combinatorial optimization problem is presented.

1.2.1 Sets and Parameters

The foundation of mathematical modeling lies in the definition of sets and parameters. A *set* S is an (unordered) collection of elements of the same type. The type of the elements can be rather general, ranging from integers (e.g., $S = \{1, 2, 5, 7\}$) to rational coordinate pairs (e.g., $S = \{(52.3, 7.1), (58.7, 23.1), (42.1, -5.2)\}$) to switching locations (e.g., $S = \{$Amsterdam, Berlin, Brussels, London$\}$) to band-

width capacities in an SDH (Synchronous Digital Hierarchy) network (e.g., $S = \{STM\text{-}1, STM\text{-}4, STM\text{-}16, STM\text{-}64\}$).

Throughout this book the set of all integers is denoted by $\mathbb{Z}$, the set of all non-negative integers by $\mathbb{Z}_+$. Similarly, the set of rational numbers will be denoted by $\mathbb{Q}$ and the set of rational and irrational numbers by $\mathbb{R}$, whereas $\mathbb{Q}_+$ and $\mathbb{R}_+$ denote their nonnegative subsets. The n-dimensional variants of these sets are denoted $\mathbb{Z}^n$, $\mathbb{Q}^n$, and $\mathbb{R}^n$ respectively.

The cardinality of a finite set S is denoted by $|S|$ and equals the number of elements in the set. The empty set is denoted by $\emptyset$. For a finite set S, the set denoted by 2^S denotes the collection of all possible subsets of S.

A *parameter* is an unchangeable value (integer, rational, irrational) representing a numerical input to a problem to be solved. Parameters can be stand-alone (e.g., the total investment budget or the signal-to-noise ratio) or defined for each element of a set (e.g., the cost c_s or the bandwidth b_s in Mbit/s for installing an SDH bandwidth capacity $s \in \{STM\text{-}1, STM\text{-}4, STM\text{-}16, STM\text{-}64\}$).

If we would like to associate a (nonnumerical) element of set T with every element of a set S, a *function* $f : S \mapsto T$ is defined. Hence, a nonnegative integer parameter b_s associated with every element of the set S can also be represented by a function $b : S \mapsto \mathbb{Z}_+$.

A set $S \subset \mathbb{R}^n$ with numerical elements is called *convex* if for all $x, y \in S$ and any $\lambda \in [0, 1]$, $\lambda x + (1 - \lambda) y \in S$ as well.

1.2.2 Graphs and Networks

One of the most elementary discrete structures to model networking problems are undirected and directed graphs.

1.2.2.1 Undirected Graphs

An *undirected graph*, or short *graph*, is a pair $G = (V, E)$ consisting of a set of vertices V and a set of edges E where each edge $e \in E$ is a two-element *unordered* subset of V. Hence, we also write $\{i, j\} \in E$ with $i, j \in V$. We further say that an edge $\{i, j\} \in E$ is *incident* to both i and j. Figures 1(a) and 1(b) show two famous graphs, the cycle on five vertices (denoted by C_5) and the Petersen graph.

Undirected graphs are used to model relations between entities that do not have a direction associated with them or where the direction does not play a role. In communication networks, undirected graphs are used to describe, for example, the topology of an optical fiber network, where the nodes represent the routers and an edge exists in the graph if and only if there is a direct optical fiber connection (link) between the routers (Figure 1(c) shows the Pan-European Triangular Topology graph defined by COST action 266). Another example is the modeling of potential conflicts between access points of a Wireless Local Access Network (WLAN). Here the

(a) Graph C_5 (b) Petersen graph   (c) COST 266 topology graph
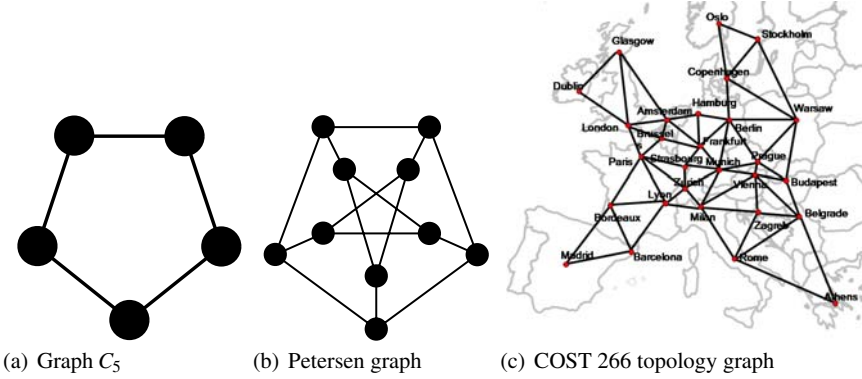
Fig. 1.1 Three undirected graphs: A cycle with five vertices and five edges, the Petersen graph with ten vertices and 15 edges, and the triangular COST 266 topology graph with 28 vertices and 61 edges

nodes represent the access points and two nodes are adjacent if and only if there are locations in the designated coverage area where the signals would interfere if both access points would be assigned the same radio frequency (see Chapter 11 for more information).

If not stated otherwise we assume that a graph is *simple* in the sense that there are no parallel edges (identical elements of E) or loops (edges of the form $\{i,i\}$).

Two distinct vertices $i, j \in V$ are called *adjacent* or *neighbors* if $\{i, j\} \in E$. This concept is extended to subsets of vertices by the function $N : 2^V \mapsto 2^V$ that assigns to every subset $S \subseteq V$ all neighboring vertices that are not part of the subset, i.e., $N(S) = \{j \in V \setminus S \mid \{i, j\} \in E, i \in S\}$. If $S = \{i\}$ we simplify notation by writing $N(i)$ instead of $N(\{i\})$. Similarly, the function $\delta : 2^V \mapsto 2^E$ assigns to every vertex subset S the edges that connect S with $V \setminus S$. The degree of a vertex is defined by the function $deg : V \mapsto \mathbb{Z}_+$ which assigns to a vertex $i \in V$ the number of adjacent edges, $deg(i) = |\delta(i)|$. If the graph G might not be clear from the context, a subscript such as deg_G is used for all three functions.

Given a graph $G = (V, E)$, we define the *complement* of G as $\bar{G} = (V, \bar{E})$ with $\bar{E} = \{\{i, j\} \mid \{i, j\} \notin E\}$.

A graph $G = (V, E)$ is called *bipartite* if the vertex set can be partitioned into two subsets V_1, V_2 such that for every edge $\{i, j\} \in E$, $i \in V_1$ and $j \in V_2$. In other words, for all $i \in V_1$, $N(i) \subseteq V_2$ and for all $i \in V_2$, $N(i) \subseteq V_1$. Bipartite graphs are therefore sometimes denoted by (V_1, V_2, E).

A graph $G = (V, E)$ is called complete if all vertices are mutually adjacent, i.e., $\{i, j\} \in E$ for all $i, j \in V$, $i \neq j$.

A graph $H = (U, F)$ is called a *subgraph* of $G = (V, E)$ if $U \subseteq V$ and $F \subseteq E$. If $F = \{\{i, j\} \in E \mid i, j \in U\}$, $G[U] = H$ is the subgraph of G *induced* by U. The subgraph of the Petersen graph induced by $\{1, 2, 3, 4, 5\}$ is C_5.

A *path* p in a graph G is a sequence $(i_0, e_1, i_1, e_2, i_2, \ldots, i_{k-1}, e_k, i_k)$ of $k+1$ vertices and k edges ($k \geq 1$) with the property that $e_j = \{i_{j-1}, i_j\}$. We write $e \in p$ and

$i \in p$ for edges and vertices that are part of the path. A path is called *simple* if no vertex appears more than once in the sequence (and hence no edge appears more than once as well). A graph G is called *connected* if there exists a path between every pair of vertices. A *component* $S \subseteq V$ of a graph $G = (V, E)$ is a subset of vertices that induces a maximally connected subgraph $G[S]$.

Two vertices $i, j \in V$ are said to be *k-edge-connected* if there exist k edge-disjoint paths between i and j in G. The *edge connectivity* $k(G)$ of a graph is the minimum over all vertex pairs of the number of edge-disjoint paths.

Similarly, two vertices $i, j \in V$ are said to be *k-vertex-connected* if there exist k vertex-disjoint paths between i and j (except for vertices i and j). The *vertex connectivity* $\ell(G)$ of a graph is the minimum over all vertex pairs of the number of vertex-disjoint paths.

A *circuit* in a graph $G = (V, E)$ is a *closed* path $(i_0, e_1, i_1, e_2, i_2, \ldots, i_{k-1}, e_k, i_k$, i.e., $i_0 = i_k$. A *cycle* is a circuit with the additional property that all vertices (and edges) except the start and end vertex are distinct.

A *tree* in a graph $G = (V, E)$ is a cycle-free connected subgraph $\mathcal{T} = (I, L)$ with $I \subseteq V$ and $L \subseteq E$. Hence, there exists a unique path between every pair of nodes $i, j \in I$. Note that in a cycle-free connected graph $|L| = |I| - 1$. If $I = V$, then $\mathcal{T}$ is a *spanning tree*.

A *stable set* or *independent set* S is a subset of the vertices such that no two vertices have an edge in common, i.e., if $i, j \in S$, then $\{i, j\} \notin E$. Stated otherwise, all vertices in the graph induced by S have degree zero. For the Petersen graph $S = \{1, 3, 7\}$ is a stable set. Since this set cannot be extended further without losing its stability, S is a *maximal* stable set. A *maximum* stable set is a stable set that is maximal and no other stable set has a higher cardinality, e.g., $\{1, 3, 9, 10\}$ is a maximum stable set.

A *clique* in a graph $G = (V, E)$ is a subset S of the vertices such that $G[S]$ is complete. Note that S is a clique in G if and only if S is a stable set in $\bar{G}$.

A *matching* M is a subset of the edges such that no two edges have a vertex in common, i.e., if $e, f \in M$, then $e \cap f = \emptyset$ (note that edges are sets of two elements). Stated otherwise, all vertices in the subgraph (V, M) have degree at most 1. For the Petersen graph, a (maximum) matching is given by $\{\{1, 2\}, \{3, 4\}, \{6, 8\}, \{7, 9\}\}$.

For an arbitrary parameter $b_i \in \mathbb{Q}$ for all $i \in V$, we define the cumulative weight function $b : 2^V \mapsto \mathbb{Q}$ as $b(S) = \sum_{i \in S} b_i$. Similarly, for an arbitrary parameter $c_e \in \mathbb{Q}$ defined for all $e \in E$, we define the cumulative weight function $c : 2^E \mapsto \mathbb{Q}$ as $c(L) = \sum_{e \in L} c_e$.

1.2.2.2 Directed Graphs

A *directed graph*, or *digraph*, is a pair $D = (V, A)$ consisting of a set of vertices V and a set of arcs A where each arc $a \in A$ is a two-element *ordered* subset of V. Hence, we also write $(i, j) \in A$. Digraphs are used in those situations where the direction of the relation is of importance, for example, in communication networks in the modeling of a traffic flow from a source node to a sink node, where it is of
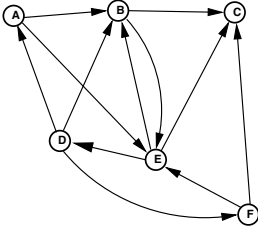
Fig. 1.2 A directed graph with seven vertices and 12 arcs

importance to know in which direction the signal is transported between the routers. Another example is the connection of wireless devices with variable transmission power. Here, an arc (i, j) exists if and only if j represents a device that is within the transmission reach of device i. Since each device has its own power control, device i is not automatically in reach of device j if j is in reach of i (see Chapter 13 for more on this application).

For digraphs, we distinguish between arcs *coming in* to a vertex $i \in V$ and arcs *going out* i. The functions $N^- : V \mapsto 2^V$, $\delta^- : V \mapsto 2^A$, $deg^- : V \mapsto \mathbb{Z}_+$ (or $N^+ : V \mapsto 2^V$, $\delta^+ : V \mapsto 2^A$, $deg^+ : V \mapsto \mathbb{Z}_+$) associate with every vertex $i \in V$ the set of incoming neighbors, arcs, and degree (or outgoing neighbors, arcs, and degree).

A (*directed*) *path* p in a digraph D is a sequence $(i_0, a_1, i_1, a_2, i_2, \ldots, i_{k-1}, a_k, i_k)$ of $k + 1$ vertices and k arcs ($k \geq 1$) with the property that $a_j = (i_{j-1}, i_j)$. We denote $a \in p$ if an arc $a \in A$ is part of the path; similarly, $i \in p$. Again, a path is called *simple* if vertices are not repeated in the sequence.

A digraph is called *strongly connected* if there exists a path from any vertex to any other vertex. A (*directed*) *cycle* is a simple directed path with $i_0 = i_k$. A digraph is called a *directed acyclic graph* or *DAG* if it does not contain directed cycles.

An *arborescence* is a digraph with the property that there is a vertex $v \in V$ such that there is exactly one directed path from v to every other vertex $u \in V$. The vertex v is called the *root* of the arborescence. Stated otherwise, an arborescence is a directed rooted tree with all arcs directed away from the root.

1.2.3 Mathematical Problems

For our purpose, a mathematical problem is the assignment $x : S \mapsto \mathbb{R}$ of values to all elements of a set S such that all constraints are satisfied. The values x_i, $i \in S$, are known as the variables of the problem. Let $n = |S|$. The constraints can be defined by functions $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ where a solution x is feasible if and only if $f_i(x) \geq 0$ for all $i = 1, \ldots, m$. The functions f_i can be defined in many different ways, from linear to highly complex.

If the goal of the mathematical problem is to find a feasible solution that max-imizes or minimizes a further function $g : \mathbb{R}^n \mapsto \mathbb{R}$, we speak of a *mathematical*

optimization problem. Hence, the general form of such a (maximization) problem is

$$\max \quad g(x) \tag{1.1a}$$
$$\text{s.t.} \quad f_i(x) \geq 0 \qquad\qquad\qquad i = 1,\dots,m \tag{1.1b}$$

Depending on the type of constraint functions, one can identify combinatorial or nonlinear optimization problems.

1.2.3.1 Combinatorial Optimization Problems

Many problems in the area of communication problems can be described as a combinatorial optimization problem. A *combinatorial optimization problem* consists of three components, a finite ground set E, a weight function $w : E \mapsto \mathbb{Z}$, and a family $\mathscr{F}$ of subsets of E. The ground set is chosen in such a way that it allows for encoding of and distinguishing between both feasible and infeasible solutions by selecting elements of the ground set. The family $\mathscr{F}$ describes all feasible solutions, and hence $2^E \setminus \mathscr{F}$ describes all infeasible solutions. The weight function is used to determine the value of a solution. For a set $E' \subseteq E$, the solution value is the cumulative weight of the elements, i.e., $w(E') = \sum_{e \in E'} w_e$. The goal of a combinatorial optimization problem is to find the best feasible solution $E' \in \mathscr{F}$, i.e., the one with minimum (or maximum) value.

The set of feasible solutions $\mathscr{F}$ can be very large and therefore is usually only given implicitly, i.e., a set of rules to determine whether or not a subset E' is feasible.

An example of an implicitly defined set of feasible solutions is the following: Given an undirected graph $G = (V,E)$, let $\mathscr{F} = \{E' \subseteq E : e \cap f = \emptyset \quad \forall e, f \in E'\}$, i.e., a subset of the edges describes a feasible solution if and only if they have no vertex in common. Such subsets are known as *matchings* (cf. Section 1.2.2.1).

Another example is the *maximum weighted independent set* in a graph $G = (V,E)$. This time the ground set is V (instead of E) and a vertex weight c_v is defined for all vertices $i \in V$. A subset $S \in \mathscr{F}$ if and only if they form an independent set, i.e., all vertices $G[S]$ have degree 0. Many more examples can be found in Section 1.5.

In many cases, the definition of feasible solutions to a combinatorial optimization problem as subsets of a ground set E is not convenient, in the sense that E should contain many copies of a certain element. For example, if we would like to install fibers between two locations, normally multiple fibers can be installed. This would imply that the ground set contains one element for every possibly installed fiber. Alternatively, the ground set can be defined as having only one element representing the fibers between the two locations, but this element can be selected multiple times in a feasible solution. Hence, E' is not anymore a subset of E, but a multi-set.

Within the framework of constraints $f_i(x)$ and an objective function $g(x)$, each solution is an assignment of 0s and 1s to the variables x_e, $e \in E$. The objective $g(x) = \sum_{e \in E} w_e x_e$ is a linear function of the variables, whereas the constraints $f_i(x)$

have to be defined in such a way that $f_i(x) \geq 0$ if and only if x defines a feasible solution (including the constraints $x_e \in \{0,1\}$ for all $e \in E$). This model can be easily extended to general integer values for x_e to represent multi-sets.

1.2.3.2 Continuous and Nonlinear Optimization Problems

Not every communication networking problem can be easily described as a combinatorial optimization problem. If decisions can be taken within a continuous spectrum of possibilities, discrete or combinatorial choices do not represent the full scale of solutions. Also the impact of certain decisions might have nonlinear effects.

If all constraint functions $f_i(x)$ and the objective $g(x)$ are linear, the problem is defined as a linear optimization problem. If at least one of these functions is nonlinear, a *nonlinear optimization* problem has to be solved. In Chapter 6, network loss models for optical burst switching are modeled with nonlinear functions, whereas in Chapter 11 a hyperbolic (hence, nonlinear) objective function is used to model the efficiency of a wireless local access network. Note that the requirements that variables x must be assigned discrete values are nonlinear functions as well.

1.2.4 Distributed Problems

In emerging applications like ad hoc wireless networking or sensor networks, centralized decisions are not favored or are even impossible due to the decentralized nature of the decision making process. In such cases, solving a mathematical optimization problem taking into account all possibilities of the decentralized units might not be implementable, and hence the decentralized or distributed problem has to be studied.

Classical combinatorial problems are *centralized*, i.e., there is a central controlling unit having complete knowledge of the input and an ability to implement decisions. In a *distributed system*, however, there is not a central unit, but many *autonomous* units (or processors), each having limited *local* knowledge of the system. Hence, decisions have to be taken by the autonomous units in a decentralized way. To enhance decision making, a processor can communicate with other processors, sometimes only in the local vicinity (modeled by a graph).

The aim of a *distributed algorithm* is, e.g., to enable communication services (routing in a wireless meshed network), to maintain control structures (backbone topology in a mobile ad hoc network), or to control resources (load balancing of processors). The quality of a distributed algorithm is usually measured by its *time complexity* and its *communication complexity*. The time complexity is measured as the number of communication rounds needed to realize the purpose of the algorithm. The communication complexity is measured as the total number of messages or volume sent by the algorithm.

The application of distributed algorithms to broadcasting is the topic of Chapter 12. We refer to [72, 78, 87] for further information on distributed computing and algorithms.

1.2.5 Online Decision Problems

In *online decision problems* not all information needed to determine the guaranteed optimal solution is available at the time of decision making. In such a case, one has to make a decision before complete knowledge of the problem input becomes available. The aim in such a situation is to make a decision that is *best* whatever the unknown input will be. The *competitive ratio* measures the quality of an algorithm in comparison to the *off-line* optimal solution, i.e., the optimal solution if the complete input were known at the time of decision.

In an online optimization problem, the missing input is modeled as a sequence of events that are unveiled one at a time. An algorithm has to be developed that reacts to the events without knowledge of further events in the sequence. The set $\mathscr{I}$ represents all considered input sequences. If for an input $I \in \mathscr{I}$, we represent the off-line optimal solution value with $OPT(I)$ and the solution value of an online algorithm A with $A(I)$, the competitive ratio is defined as

$$c(A) := \max_{I \in \mathscr{I}} \frac{A(I)}{OPT(I)} .$$

Hence, the competitive ratio measures the worst-case performance of the algorithm. A problem is called c-competitive if there exists an algorithm A with competitive ratio c. It is sometimes possible to show for a problem that there is no constant $c \geq 1$ such that there exists an algorithm that is c-competitive.

One of the classical examples of online optimization is the *paging problem*. In the paging problem, we consider two levels of computer memory, the *slow* memory containing N pages $p_1, \ldots, p_N$ and the *fast* memory (cache) that can store an arbitrary subset of $k < N$ pages. Pages loaded in the cache can be accessed directly when requested (known as a *cache hit*), whereas the other pages first have to be loaded from the slow memory into the cache (known as a *cache miss*). If the cache is fully loaded and another page is requested, one of the pages in the cache must be removed. The problem is to find an algorithm that minimizes the number of cache misses, without knowledge of which pages are requested in the future.

Given a sequence $r_1, \ldots, r_n$ of page requests, we have to decide for each cache miss which page to remove. In the off-line setting, i.e., the complete sequence of requests is known in advance, the *Longest-Forward-Distance algorithm* [13] provides an optimal solution: At every cache miss, remove the page whose next access is most distant in the future.

In the online setting, several algorithms have been proposed, such as *First-In-First-Out*, *Last-In-First-Out*, and *Least-Recently-Used*. The latter removes the page

that has been in the cache the longest time without being requested. It can be shown that First-In-First-Out and Least-Recently-Used are k-competitive [84]. Moreover, it has been proved that no *deterministic* algorithm (i.e., an algorithm that does not include randomized decisions) exists that is c-competitive with $c < k$ [84].

Algorithms with a better performance can be obtained by using *randomized* decisions. In particular, the random *marking* algorithm of [47] has a competitive ratio of $2H_k$, where $H_k = 1 + \frac{1}{2} + \ldots + \frac{1}{k}$ is the kth Harmonic number (note that $H_k \leq 1 + \ln k$. The algorithm works in phases as follows. Initially all pages in the cache are unmarked. A phase is ended as soon as all pages in the cache are marked. In this case, all pages are unmarked and a new phase begins. This way, we always have at least one page unmarked upon the arrival of a request for a page p. If page p is in the cache, it is marked. If page p is not in the cache, we randomly choose an unmarked page in the cache to be replaced by p, and p is marked.

We refer to [52] for further reading in the area of online optimization. Applications of online optimization in communication networks can be found in Chapters 2 and 10.

1.3 Computational Complexity

Computational Complexity theory is the science that studies the computational resources (time, memory, etc.) needed to solve computational problems. It is especially concerned with the distinction between *tractable* problems, that can be solved with reasonable amount of resources, and *intractable* problems, that are beyond the power of existing, or conceivable, computers.

Obviously the computational resources needed to solve a problem depend on the size of the problem input data. A problem is considered *efficiently tractable* if the resources needed grow at most as a polynomial function in terms of the input data, and is considered not efficiently computable otherwise.

A *complexity class* is the set of problems solvable by a particular computational model under a given set of resource constraints. Therefore, if we focus on time as the main resource, or equivalently, the number of elementary computer operations required to solve the problem, we define a first complexity class as follows.

P is the class of decision problems that can be solved in polynomial time on a *deterministic effective computing system* (ECS). Loosely speaking, all computing machines that currently exist in the real world are deterministic ECSs. So, P is the class of problems that can be computed in polynomial time on real computers. A *decision problem* is a problem for which the solution is a "yes" or "no" answer.

NP is the class of decision problems that can be solved in polynomial time on *non-deterministic* ECSs. A *non-deterministic machine* is a machine which can execute programs in a way where, whenever there are multiple choices, rather than iterate through them one at a time it can follow all choices or paths at the same time, and the computation will succeed if any of those paths succeed; if multiple paths lead to success, one of them will be selected by some unspecified mechanism; we

usually say that it will pick the first path to lead to a successful or "yes" result. The idea of non-determinism in effective computing systems can be explained without entering into details of non-deterministic computing machines (the reader is referred to [8, 50] for a more detailed explanation). A decision problem whose "yes" solution can be computed in polynomial time on a non-deterministic machine is equivalent to a problem where a proposed "yes" solution can be verified as correct in polynomial time on a deterministic machine. You can specify a non-deterministic machine that guesses one solution, following all of those paths at once, and returning a result whenever it finds a solution that it can verify is correct. So, if you can check a solution deterministically in polynomial time (not produce a solution, but just verify that the solution is correct), then the problem is in NP.

The distinction can become much clearer with an example. A classic problem is the *subset sum problem*. In the subset sum problem, an arbitrary set of integers is given. The question is whether there exists a nonempty subset of values in the set whose sum is 0? It should be pretty obvious that checking a solution is in P: a solution is a list of integers whose maximum length is the size of the entire set; to check a potential solution, add the values in the solution, and see if the result is 0. The computational effort of this procedure is $\mathcal{O}(n)$ where n is the number of values in the set. But finding a solution is hard. The solution could be any subset of any size larger than 0; for a set of n elements, there are $2^n - 1$ such subsets. Even if you use *clever tricks* to reduce the number of possible solutions, you are still in exponential territory in the worse case. But you can non-deterministically guess a solution and test it in linear time; but no one has found any way of producing a correct solution deterministically in less than $\Theta(2^n)$ steps.

One of the great unsolved problems in theoretical computer science is does P = NP? That is, is the set of problems that can be solved in polynomial time on a non-deterministic machine the same as the set of problems that can be solved in polynomial time on a deterministic machine? It is clear that that P ⊆ NP, that is, that all problems that can be solved in polynomial time on a deterministic machine can also be solved in polynomial time on a non-deterministic machine. Although it is a commonly accepted hypothesis that P ≠ NP no one has been able to prove it to date.

Within NP, there is a set of particularly interesting problems which are called NP-complete. The idea of an NP-complete problem is that it is *one of the hardest problems in* NP or, in other words, is one where we can prove that if there is a P-time computation that solves the problem, it would mean that there was a P-time solution for every problem in NP, and thus P = NP.

How do we show that a given problem is NP-complete? NP-completeness is based on the idea of *problem reduction*. Given two problems S and T for which it can be shown that any instance of S can be transformed into an instance of T in polynomial time, it is said that S is *polynomial-time reducible* to T. Therefore, if an efficient algorithm to solve problem T is known, this algorithm can also be used to solve problem S. It can be seen as S *is easier than* T.

Once we know a problem T which is NP-complete, then for any other problem U, if we can show that T is polynomial-time reducible to U, then U must be NP-

complete as well. If we can reduce T to U, but we do not know how to reduce U to T or we do not know if $U \in$ NP, then we do not just say that U is NP-complete; we say that it is NP-hard.

There are a lot of problems which have been proved NP-complete. So, given a new problem, there are a lot of different NP-complete problems that can be used as a springboard for proving that the new problem is also NP-complete.

Most NP-completeness proofs are ultimately built on top of the NP-completeness proof of one fundamental problem, whose nature makes it particularly appropriate as a universal reduction target and it is also the first problem that was proved to be NP-complete. It is called the propositional satisfaction problem (a.k.a. SAT or satisfiability), which was shown to be NP-complete via a rather abstract model (Cook Theorem [50]). For any other problem, if we can show that we can translate any instance of a SAT problem to an instance of some other problem in polynomial time, then that other problem must also be NP-complete. And SAT (or one of its simpler variations, 3-SAT) is particularly easy to work with, and it is easy to show how to translate instances of SAT to instances of other problems.

Let us see an example. A vertex cover of an undirected graph $G = (V, E)$ is a subset V' of the vertices of the graph such that every edge in G has an endpoint in V', i.e., $\forall (u, v) \in E : u \in V' \vee v \in V'$.

The vertex cover problem is the optimization problem of finding a vertex cover of minimum size in a graph. The problem can also be stated as a decision problem: Given a graph G and a positive integer k, is there a vertex cover of size k or less for G?

Vertex cover is closely related to the Independent Set problem: V' is a vertex cover if and only if its complement, $V \setminus V'$, is an independent set. It follows that a graph with n vertices has a vertex cover of size k if and only if the graph has an independent set of size $n - k$. Equivalently a graph $G = (V, E)$ with n vertices has a vertex cover of size k if and only if the complementary graph $\overline{G} = (V, \overline{E})$ have a clique of size $n - k$. This equivalence shows a trivial polynomial reduction from clique to vertex cover. Since clique (does a graph has a clique of given size?) is an NP-complete problem (see [50]), we have shown the NP-completeness of vertex cover.

1.4 Combinatorial Optimization Methods

Given a (classical) combinatorial optimization problem by its three components, the ground set E, (an implicit definition of) the feasible solutions $\mathscr{F}$, and a weight function $w : E \mapsto \mathbb{Z}$, the problem can be formulated as *mathematical optimization problem* by introducing *decision variables* for all elements of the ground set E. For every $e \in E$ the decision variable x_e can take the values 0 or 1 (and is therefore called *binary*) indicating whether e is chosen (1) or not (0) in the optimal solution. Hence, the *objective* can be written as $\sum_{e \in E} w_e x_e$. We further define $x_{E'} = (x_e)_{e \in E}$ to be the *incidence vector* for a subset $E' \subseteq E$, i.e., $x_e = 1$ if $e \in E'$ and 0 otherwise.

By defining $X = \{x_{E'} \mid E' \in \mathscr{F}\}$, we now can write the problem as

$$\max \left\{ w^T x \mid x \in X \right\} . \tag{1.2}$$

If convenient from the problem perspective, the above might be generalized to multi-sets to represent decisions that can be made multiple times.

To find an optimal solution for the above problem, a number of methods are available. In the following we first introduce linear programming problems and briefly describe algorithms to solve such problems. Next, we discuss how combinatorial optimization problems can be solved with linear-programming-based techniques. Alternative solution methods like graph theory, problem-specific combinatorial algorithms, approximation algorithms, and heuristics are discussed in Sections 1.4.2, 1.4.3, 1.4.4, and 1.4.5 respectively. Both linear-programming-based branch-and-bound and graph algorithms are examples of *exact algorithms*, i.e., they provide the optimal solution in the end. In contrast, heuristics and approximation algorithms provide a good but not necessarily optimal solution.

Note that the above discussion becomes more complicated as soon as more general problems are studied. A brief discussion of nonlinear optimization methods can be found in Section 1.4.6

1.4.1 Linear-Programming-Based Methods

1.4.1.1 Polyhedral Theory

Since the components of all vectors $x \in X$ are either 0 or 1, problem (1.2) is equivalent to

$$\max \left\{ w^T x \mid x \in P \right\},$$

where $P = \text{conv}(X)$ is the *convex hull* of all vectors in X, i.e.,

$$\text{conv}(X) := \left\{ y \mid \exists t \in \mathbb{Z}_+, \exists x^1, \ldots, x^t \in X, \exists \lambda \in [0,1]^t, \sum_{i=1}^{t} \lambda_i = 1, y = \sum_{i=1}^{t} \lambda_i x^i \right\}$$

A set P that can be written as the convex hull of a finite number of vectors is called a *polytope*. A set $P \subset \mathbb{R}^n$ is called a *polyhedron* if and only if there exists $m \geq 0$ such that

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\},$$

where A is an $m \times n$ matrix and $b \in \mathbb{R}^m$ (if $m = 0$, $P = \mathbb{R}^n$). A polytope is a *bounded* polyhedron. An example is given in Figure 1.3.

So, instead of an implicit description of all feasible solutions, we can give a polyhedral description of the solutions by a system $Ax \leq b$. In fact, in the case of
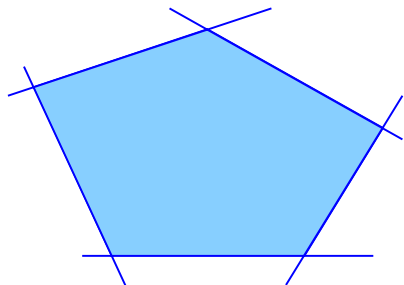
Fig. 1.3 A two-dimensional polyhedron described by a system of five inequalities

binary vectors describing the feasible solutions, the polyhedron is bounded. This fact is the key to solving combinatorial optimization problems with *linear programming* techniques.

In case the solutions are represented by multi-sets of the ground set E, a similar approach can be followed, with the difference that the solution space may be unbounded (in theory), and therefore we have to study a polyhedron instead of a polytope. Although not necessary if the objective is linear, it is usually assumed that every integer solution that can be written as a convex combination of integer *feasible* solutions is also feasible. This assumption is not needed in the case of binary vectors since binary vectors cannot be written as a convex combination of other binary vectors.

1.4.1.2 Solving Linear Programming Problems

A *linear program* (LP) is an optimization problem satisfying the following conditions

- the objective is a linear function of the variables
- all constraints are linear in the variables
- the variables can take any values (i.e., $\in \mathbb{R}$) that satisfy the constraints

By this definition, an LP can be written w. l. o. g. in its standard maximization form as follows

$$
\begin{cases}
\max\ c^T x \\
\quad \text{s.t. } Ax = b \\
\quad\quad x \geq 0
\end{cases}
\tag{1.3}
$$

with matrix $A \in \mathbb{Q}^{m \times n}$, a right-hand side $b \in \mathbb{Q}_+^m$, and arbitrary objective values $c \in \mathbb{R}^n$ (here, m is the number of rows and n the number of columns of A). Objectives to be minimized can be rewritten as a maximization problem by multiplying the coefficients by -1. Likewise for equations with negative right-hand sides. Inequalities can be written as equations by introducing slack variables. Upper bound constraints for single variables are included in the coefficient matrix (with a slack variable).

Negative lower bounds can be avoided by substituting variables, and therefore we can assume that all variables are nonnegative.

The optimal solution of a linear program (1.3) can be found by a variety of methods developed since the 1940s. A first method is *Fourier-Motzkin elimination* for solving systems of linear inequalities [82, Chapter 12], but it is computationally rather intensive. More efficient algorithms (either in theory or practice) are provided by the Simplex method and interior point methods.

Simplex Method

The *Simplex method* was developed as early as 1947 by Dantzig [36]. Assuming there exist feasible solutions to (1.3), the Simplex algorithm exploits the fact there exists an optimal solution that is an *extreme point* (vertex) of the polyhedron defined by $Ax = b$ and $x \geq 0$. Therefore, the Simplex algorithm walks along the extreme points of the polyhedron in such a way that the objective value of the sequentially considered solutions is improving. For this a first extreme point of the polyhedron has to be found. This is known as *Phase I*, whereas the walk along the extreme points of the polyhedron to an optimal solution is known as *Phase II*. Figure 1.4 illustrates the procedure for an objective and the polyhedron of Figure 1.3.
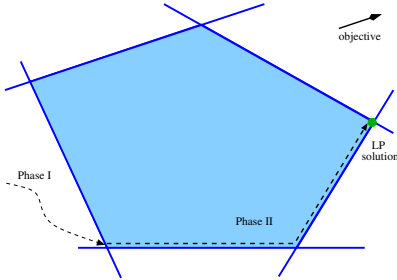
Fig. 1.4 Given an objective, at least one optimal solution is an extreme point of the polyhedron; the Simplex method uses this property by walking along the extreme points

If $b = 0$, $x = 0$ is a feasible solution of (1.3). If $b \neq 0$, this solution is infeasible and Phase I consists of setting up an auxiliary linear program for which a feasible solution can be easily found. Next, Phase II is applied to this auxiliary LP to find a feasible solution of (1.3). The auxiliary LP is obtained by introducing variables $s_i \geq 0$, $1 \leq i \leq m$. The aim is to find a solution with $s = 0$, and thus the auxiliary LP reads

$$\begin{cases} \max \ \mathbb{1}^T s \\ \text{s.t. } Ax + s = b \\ \qquad x, s \geq 0 \end{cases} \tag{1.4}$$

where $\mathbb{1}$ is the all-one vector of dimension m. It holds that (1.3) has a feasible solution if and only if the optimal solution of (1.4) has $s = 0$. A feasible solution of (1.4)

is given by $(x,s) = (0,b)$ (note $b \geq 0$ in (1.3)). So, we directly can start with Phase II for this auxiliary LP. If the optimal solution (x^*,s^*) has $s^* \neq 0$, the polyhedron defined by $Ax = b$, $x \geq 0$ is empty; otherwise, x^* is a feasible solution for (1.3).

Phase II works with *basic feasible solutions*. A basic solution to $Ax = b$ (with $n \geq m$) is obtained by setting $n - m$ variables equal to 0 (the *non-basic* variables) and solving the values of the remaining m variables (the *basic* variables), assuming that the columns for the remaining m variables are linearly independent. If all variables in a basic solution are nonnegative, it is called a basic feasible solution. It can be shown that a point in the feasible region of LP (1.3) is an extreme point if and only if it is a basic feasible solution to $Ax = b$. Now, two basic feasible solutions are said to be *adjacent* if their sets of basic variables have $m - 1$ basic variables in common.

Since an LP with a nonempty feasible region always has an optimal basic feasible solution, the solution provided by Phase I is a basic feasible solution of (1.3). Starting with this solution, the Phase II of the Simplex algorithm exchanges one basic variable for a non-basic variable in order to become a better objective value. Such an exchange is called a *pivot*. As long as the basic feasible solution is not optimal, such a pivot exists. The performance of the Simplex algorithm heavily depends on the selection of the pivot element (including so-called cycling between basic feasible solutions with the same objective value). An optimal solution is found as soon as none of the neighboring basic feasible solutions have a better objective value (we omit the technical details on how to check this condition here).

Klee and Minty [67] have shown that the Simplex algorithm might take an exponential number of pivots to find the optimal solution. In practice, however, revised and dual versions the Simplex algorithm are still considered as the best algorithms to solve linear programs [20]. For an in-depth introduction to the Simplex algorithm we refer to [29, 88].

Interior Point Methods

The worst-case exponential behavior of the Simplex algorithm has motivated research for alternative, polynomial time, algorithms to solve a linear program. Starting with the work of Karmarkar [65], so-called *interior point methods* resolved this question. Interior point methods basically do not follow a path along the extreme points of the polyhedron, but go through the interior of it. If a single optimal solution (which is an extreme point) exists, the algorithm approximates this solution, and a final rounding step will result in the optimal solution. If multiple optimal solutions exist, an interior point algorithm might approximate any convex combination of the extreme optimal solutions, and the resulting optimal solution might be different from the optimal extreme points. We refer to [88] for a detailed discussion.

Dynamic Column Generation

Some formulations of optimization problems as (integer) linear programs require a very large number of variables, typically exponential in the problem input size. An example is the set of variables representing all possible paths between two nodes in a network; see Section 1.5.2.2. In such cases, it is irrational and often impossible to generate and store all variables in the optimization software. The implicit handling of such very large sets of variables is known as *dynamic column generation*. Instead of all variables, only a small subset of the variables is generated in the beginning, and the remaining variables are not considered in the first instance. For the initial subset of variables the reduced linear program known as the *master program* is solved to optimality. Next, all remaining variables (implicitly set to 0) are checked for the capability of potentially improving the solution by adding them to the small set of explicitly handled variables. If some candidates are found, they are added to the linear program, and this is solved again. The procedure is repeated until no further candidates are found. In this case, the optimal solution is found without dealing with all variables explicitly. The problem of finding candidate variables to be included in the master program is known as the *pricing problem* and usually can be formulated as a linear or integer linear program.

1.4.1.3 Solving Mixed-Integer Linear Programming Problems

An *integer linear program* (ILP) is an optimization problem satisfying the following conditions

- the objective is a linear function of the variables
- all constraints are linear in the variables
- the variables can take only integer values (i.e. $\in \mathbb{Z}$)

By this definition, an ILP can be written w. l. o. g. in its standard maximization form as

$$\begin{cases} \max\ c^T x \\ \quad \text{s.t. } Ax = b \\ \qquad x \in \mathbb{Z}_+^n \end{cases} \tag{1.5}$$

with matrix $A \in \mathbb{Q}^{m \times n}$, a right-hand side $b \in \mathbb{Q}_+^m$, and arbitrary objective values $c \in \mathbb{R}^n$. A *mixed-integer program* (MIP) generalizes the above definition by requiring only a subset of the variables to be integer and allowing the remaining ones to take continuous values. The methods to solve ILPs and MIPs are basically the same, and therefore not further distinguished in this chapter.

For the purpose of our exposition, it is more convenient to rewrite the formulation (1.5) to a system of linear inequalities:

$$\begin{cases} \max\ c^T x \\ \quad \text{s.t. } Ax \le b \\ \qquad x \in \mathbb{Z}_+^n \end{cases} \tag{1.6}$$

The *linear relaxation* of (1.6) is obtained by replacing the integrality constraints by nonnegativity constraints. Figure 1.5 illustrates for a given system of linear inequalities the integer-feasible solutions (i.e., the grid of points) as well as the LP relaxation (the polytope).
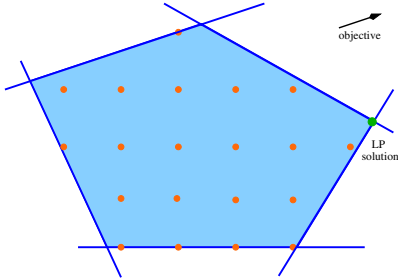
Fig. 1.5 Integer-feasible solutions and the LP relaxation for a system of linear inequalities

Branch-and-Bound

The most common method to solve ILPs is *branch-and-bound* (B&B). In a B&B algorithm, the linear programming relaxation of (1.6) is solved multiple times, each time with other bounds on the variables. Let z be the optimal solution value of (1.6). The algorithm starts with solving the LP relaxation, returning a solution $\hat{x}$. Since the LP relaxation contains all integer-feasible points (cf. Figure 1.5), the value of the LP relaxation $z_{LP}^0 = c^T \hat{x}$ is an upper bound on z. If $\hat{x} \in \mathbb{Z}_+^n$, the solution is also valid for (1.6), and hence we found an optimal solution. If $\hat{x} \notin \mathbb{Z}_+^n$, there is at least one fractional variable $\hat{x}_i$. In an optimal solution, either $x_i \leq \lfloor \hat{x}_i \rfloor$ or $x_i \geq \lceil \hat{x}_i \rceil$. Therefore, we perform *branching* on x_i: we replace the LP relaxation with two new subproblems, one with the variable bound $x_i \leq \lfloor \hat{x}_i \rfloor$ and one with the variable bound $x_i \geq \lceil \hat{x}_i \rceil$; see Figure 1.6 for illustration.
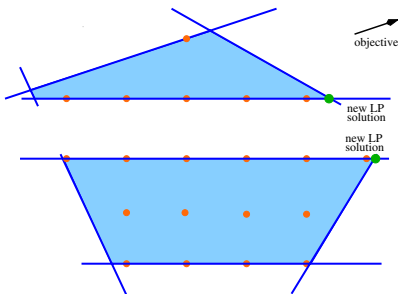
Fig. 1.6 Branching on a fractional variable: two subproblems provide new optimal LP solutions

Every optimal solution of (1.6) is in either of the new polyhedra, and thus $z = \max\{z^1, z^2\}$ where z^1 and z^2 are the optimal (integer) solution values of the subproblems respectively. Moreover, $z \leq \max\{z_{LP}^1, z_{LP}^2\} \leq z_{LP}^0$ *bounds* the optimal value from above, where z_{LP}^1 and z_{LP}^2 are the values of the corresponding LP relaxations. In case the LP relaxation of the first (second) subproblem is integral, $z \geq z_{LP}^1 = z^1$ ($z \geq z_{LP}^2 = z^2$), and we can *bound* the optimal value from below. The procedure can be repeated recursively for the subproblems as long as the two bounds are not equal. For further details, we refer to [82, 92] or to any one of the many textbooks in Operations Research.

Polyhedral Combinatorics

Polyhedral combinatorics is the study of the structure of the polytope (polyhedron) described by the integer solutions of (1.6). Given the set of integer solutions, we can define P as the convex hull of these points; see Figure 1.7. If we can derive a system of linear inequalities describing this polytope, (1.6) can be solved by the Simplex Algorithm (or interior point methods, possibly with a slight permutation of the objective) since every extreme point is integral by definition.
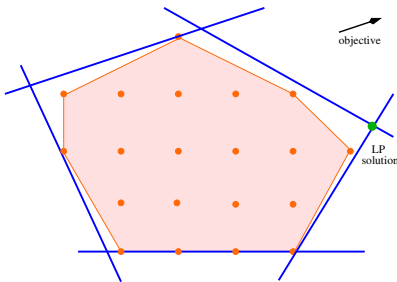
Fig. 1.7 The convex hull of integer solutions can be described by a system of linear inequalities

An inequality $ax \leq a_0$ is called *valid* if $ax^* \leq a_0$ for all $x^* \in P$. Let $F(a, a_0) = \{x \in P : ax = a_0\}$ be the set of polytope points that satisfy the valid inequality with equality. If $F \neq \emptyset$ it describes a face of P. A valid inequality is *facet-defining* if $F(a, a_0)$ is maximal, i.e., there is no other valid inequality $dx \leq d_0$ with $F(a, a_0) \subset F(d, d_0)$. Stated otherwise, the intersection of the polytope and a facet-defining valid inequality forms a face of the polytope of highest possible dimension (without having $F = P$); see Figure 1.8.

A valid inequality is also called a *cutting plane* as it might cut off a fractional solution from the integer solutions. A cutting plane algorithm follows this procedure. First, the LP relaxation is solved. Next, if the LP solution $\hat{x}$ is not integral, there exists a valid inequality $ax \leq a_0$ that is *violated* by $\hat{x}$, i.e., $a\hat{x} > a_0$. This inequality is added to the system of inequalities defining the LP relaxation, and the enlarged LP

(a) Valid inequality   (b) Facet of polyhedron
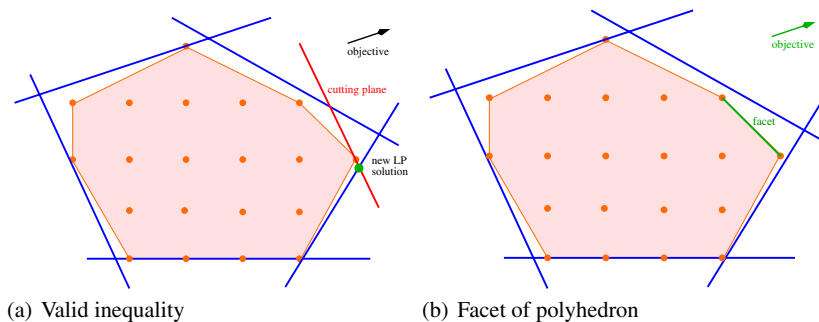
Fig. 1.8 A cutting plane is a valid inequality that separates the LP relaxation from the integer-feasible solutions; the strongest cutting planes are facets of the integer polyhedron

is resolved. This procedure is repeated until the optimal LP solution is integral. The search for a violated inequality is known as the *separation algorithm*.

If all facet-defining inequalities of *P* are known (a so-called *complete description*), the described algorithm is guaranteed to find an optimal solution of (1.6). However, since interior point methods find an optimal solution in polynomial time, deriving a complete description of the convex hull of integer points is at least as difficult as solving the optimization problem itself; i.e., if the problem is NP-hard, deriving a complete description of size polynomial in the input is not possible, unless P = NP. Stated differently, if a complete description of the polytope can be identified, the number of inequalities must be of exponential size, unless P = NP. Further, given a fractional solution $\hat{x}$ of (1.6), determining a violated valid inequality is, in general, as hard as solving the optimization problem itself. This theorem is known as *separation = optimization* [53, 54]. For further details we refer to [55]. A compact presentation of polyhedral techniques for combinatorial optimization can be found in [1, 2].

Branch-and-Cut

If not all facet-defining inequalities of *P* are known, the cutting plane algorithm might not be able to find a violated valid inequality given an LP solution $\hat{x}$. Or it may take too long to find a violated inequality. In such cases, the cutting plane algorithm ends with a fractional solution and a (hopefully) improved upper bound. Branching on one of the fractional variables is now an option. For each of the subproblems, the cutting plane algorithm can be restarted to improve the bounds further. This combination of branch-and-bound and cutting planes is known as *branch-and-cut*.

All modern integer linear programming solvers exploit a branch-and-cut algorithm, where general purpose cutting planes like *Chvátal-Gomory cuts* and *clique inequalities* are separated. Products like ILOG CPLEX [64] and SCIP [5] allow for the addition of problem-specific cutting planes by the user.

Branch-and-Price

The case where the ILP (1.5) has a very large number of variables can be handled similarly to the linear programming case, using a master and pricing problem. However, this time such a dynamic column generation approach has to be interwoven with the B&B algorithm. The resulting algorithm is known as *branch-and-price* (B&P). Special care has to be taken to solve a problem with B&P since branching on variables might be in conflict with the column generation, i.e., if we branch on a variable to be bounded from above (i.e., $x_i \leq \lfloor \hat{x}_i \rfloor$, we should prevent the same variable from being generated by the column generation, since otherwise the same subproblem as before is solved. For further information on B&P, or the integration of cutting planes, known as *branch-and-cut-and-price* (B&C&P), we refer to [11].

Further Reading

More on (mixed) integer linear programming methods and polyhedral theory can be found in [76, 82, 92]. Recent progress can be found in [21], whereas [31] provides a nice historical view.

1.4.2 Graph Theory

Graph theoreticians study the properties of particular (classes of) graphs and search for equivalent characterizations among them. In many cases, graph-theoretical models and concepts turned out to be very relevant for communication networks, e.g., for *interconnection networks* or GSM frequency planning (see Section 1.5.5.1). But networking problems have also turned into graph-theoretical questions and answers. For example, the study of non-blocking multistage switching networks for telephony networks has resulted in the so-called *Clos network* [30]. Here are a few other examples taken from [35]: Design of dense networks [15, 25], traffic congestion (forwarding index) [28, 61], broadcasting algorithms and dissemination (gossiping) of information [63], fault tolerance (surviving route graph [40] or connectivity [45]).

In the last two decades, the development of optical networking technologies has required the solution of classical graph-theoretical problems. As observed by many authors, the wavelength assignment problem in an all-optical network is in essence equivalent to the vertex coloring problem in its conflict graph, e.g., [46]. Another example is networks based on the Optical Transpose Interconnection System (OTIS) architecture [73]. It has been shown in [34] that they have a topology which is highly related to the very well-known families of directed graphs called *Kautz* and *De Bruijn graphs*. Those families have been proposed many times as topologies for interconnection networks due to their good properties.

A number of high-quality textbooks on graph theory exist (for example, [91], some of them available online, e.g., [22, 38]). For the theory of digraphs, we refer to [9].

1.4.3 Combinatorial Algorithms

Many combinatorial optimization problems can be solved by specialized algorithms in polynomial time. This is in particular true for problems with an underlying (directed) graph structure. Such algorithms are called *combinatorial algorithms*. A combinatorial algorithm takes the problem parameters as input and outputs the optimal solution for the problem at hand. We distinguish a number of different cases.

A *greedy algorithm* is an algorithm that constructs a solution by irrevocably selecting components of the solution, i.e., once a component of the solution is selected, it will not be removed anymore from the solution. An example of a greedy algorithm is the Dijkstra-Prim algorithm for computing a minimum spanning tree; cf. Section 1.5.1.1.

A *dynamic programming* algorithm uses optimal solutions to subproblems of the original problem to compute the optimal solutions for the original problem. An example for such an algorithm is the dynamic programming algorithm for the knapsack problem, where knapsack problems with fewer items and smaller volumes are solved recursively; see [74].

Dijkstra's algorithm for the shortest path problem (see Section 1.5.2.1) is neither a greedy algorithm nor a dynamic programming algorithm since solution values are updated before the optimal solution (of subproblems) is found. Such combinatorial algorithms exist for many well-structured problems. Schrijver [83] is a great source for problems (and the algorithms of course) that can be solved in this way. All these problems have in common that they can be solved in polynomial time and thus belong to the class P.

1.4.4 Approximation Algorithms

If a problem is not known to be a member of class P but rather is known to be NP-complete, there still may exist algorithms that provide a solution in polynomial time. However, such algorithms do not guarantee that the solution is optimal.

The class APX is the set of NPO problems (optimization problems whose decision version is in NP) that allow polynomial-time *approximation algorithms* with *approximation ratio* bounded by a constant. An approximation algorithm is called an α-approximation algorithm for some constant α if it can be proved that the solution that the algorithm finds is at most α times worse than the optimal solution. Here, α is called the *approximation ratio*. Depending on whether the problem is a minimization or a maximization problem, this can either denote α times larger or

α times smaller, respectively. For example, the vertex cover problem and traveling salesman problem with triangle inequality each have simple 2-approximation algorithms. In contrast to that it is proved that the traveling salesman problem with arbitrary edge lengths cannot be approximated with an approximation ratio bounded by a constant, unless the Hamiltonian path problem can be solved in polynomial time.

If there is a polynomial-time algorithm to solve a problem within every fixed constant α (with running time dependent on α), then the problem is said to have a *polynomial-time approximation scheme* (PTAS). Unless P = NP, it can be shown that there are problems that are in APX but not in PTAS; that is, problems that can be approximated within some constant factor, but not every constant factor. A problem is said to be APX-hard if there is a PTAS reduction from every problem in APX to that problem. To say a problem is APX-hard is generally bad news, because it denies the existence of a PTAS, which is the most useful sort of approximation algorithm. For further details, we refer to [8].

1.4.5 Heuristics Without Solution Guarantee

A further class of algorithms provides a solution without any guarantee on the quality. In such a case we speak about a *heuristic*. Some heuristics generate a solution from scratch and are therefore called *constructive heuristics*. A *local search algorithm* takes a solution as part of the input and tries to improve this solution, e.g., by exchange operations. A good source on local search algorithms is [4].

More computationally intensive algorithms are known under the collective term *metaheuristics*, including *genetic algorithms*, *tabu search*, *simulated annealing*, *artificial neural network*, and *ant colony optimization*. We again refer to [4] for a review of those methods.

1.4.6 Nonlinear Programming

For *nonlinear programming* problems the solution methodology is less standardized than for linear programming problems. This is mainly due to the fact that one has to distinguish between *local optimal* and *global optimal solutions*. A local optimal solution $\hat{x}$ is a solution such that for any small perturbation ε, $\hat{x} + \varepsilon$ is either infeasible or its objective value is worse. However, it does not guarantee that there is no other solution y with a better objective value. For the global optimal solution it holds that there does not exist any other feasible solution with a better objective value.

For linear programming problems (more precisely, *convex optimization problems*) each local optimal solution is also globally optimal, which allows methods like the Simplex algorithm to work. As this is not the case for general nonlinear programming problems (non-convex to be more precise), methods usually only guaran-

tee a local optimal solution as output. Valuable sources for the theory of nonlinear programming are [12, 16].

1.5 Selected Classical Applications in Communication Networks

In this section a few classical applications of graphs and algorithms for communication networks are discussed. First, the design of network topologies is discussed, as well as are some routing problems such as the shortest path and minimum cost flow. In Section 1.5.3 basic versions of the network planning/design problem are introduced. We elaborate on an application to optical network design in Section 1.5.4, before two classical applications in wireless networks are discussed in the final Subsection 1.5.5.

1.5.1 Design of Network Topologies

The design of network topologies has a long tradition of providing combinatorial optimization problems, starting with the design of tree and ring networks to meshed network structures that guarantee multiple node- or link-disjoint paths between every pair of network nodes.

1.5.1.1 Design of Tree Topologies

To enable communication between a pair of network nodes, either direct or via other network nodes (serving as switches), there must exist at least one path between them. In topology network design problems costs are associated with the usage of a potential link between two network nodes. These costs can represent various real cost factors such as the digging of a cable trunk, the leasing of a virtual connection, the laying of a cable in an office building, or the installation of a configuration of a directed radio link. Capacity of the link does not play a role as we only consider the possibility to communicate and not the amount of communication (cf. Sections 1.5.2 and 1.5.3 for these issues).

Depending on the technology, a link may be directed or undirected. In this chapter we only consider the undirected version of this problem; we refer to [83, Chapter 52] for the directed version. We abstract from the practical application by the introduction of two graphs: The graph $G = (V, E)$ describes all potential connections between the networks nodes V. We associate with every edge $ij \in E$ a cost value $\kappa_{ij} \in \mathbb{Q}$ denoting the installation cost of a link between the network nodes. W. l. o. g. we can assume that G is complete by $\kappa_{ij} = \infty$ for all not yet existing edges $ij \notin E$. A minimum cost subset of the edges $L \subseteq E$ has to be selected as network topology. A second graph $H = (U, F)$ with $U \subseteq V$ encodes all required communication paths.

An edge $ij \in F$ exists if and only if the topology solution should contain a path between the two network nodes. Three cases are distinguished in the following:

1. $U \equiv V$ and H is complete – the minimum spanning tree problem,
2. $U \subset V$ and H is complete – the minimum Steiner tree problem, and
3. H is not a complete graph – the minimum Steiner network problem.

Each of the cases is illustrated by an (artificial) example from communication networking.

Minimum Spanning Tree Problem

Application 1.1 *In a new office building, a Wireless Local Area Network (WLAN) has to be installed. For this a number of so-called access points (APs) have been determined that allow for wireless coverage of the building. The APs have to be interconnected via a wired Ethernet backbone. The cost of a direct wired connection between two APs i and j is denoted by $\kappa_{ij} \in \mathbb{Q}$. One has to design a minimum cost wired network topology that enables communication between all APs.*

In this application no other connection points than the APs exist. Hence, V denotes the set of APs and $S = V$. This problem is known as the *minimum cost spanning tree* problem. If a connecting path is required between all vertex pairs $i, j \in V$, at least $|V| - 1$ edges have to be selected and the resulting subgraph (V, L) must be cycle-free (otherwise certain vertices are not connected). Such subgraphs are exactly the spanning trees in G. Thus, in this case the optimal topology is a minimum cost spanning tree $\mathcal{T} = (V, L)$.

A minimum cost spanning tree in a weighted graph $G = (V, E)$ can be found with for example the *Dijkstra-Prim algorithm* [39, 81]; see Algorithm 1.1. The algorithm selects repeatedly the minimum cost edge extending the current tree, starting with a single vertex, and ending with a tree spanning all vertices.

Algorithm 1.1 Dijkstra-Prim algorithm to determine the minimum cost spanning tree $\mathcal{T} = (V, L)$ in a graph $G = (V, E)$ with edge cost κ_e.

Let $L := \emptyset$
Let $S := \{i\}$ for some arbitrary $i \in V$
while $S \neq V$ **do**
    Let $e = \arg\min\{\kappa_{ij} \mid ij \in E, i \in S, j \in V \setminus S\}$
    $L := L \cup \{e\}$
    $S := S \cup \{j\}$ with $e = \{i, j\}, i \in S$
return $\mathcal{T} = (V, L)$ with cost $\kappa(L)$

Optimality of the final spanning tree can be proved by an exchange argument: Assume $\mathcal{T}$ is not optimal and let $\mathcal{T}' = (V, L')$ be an optimal spanning tree with $|L \cap L'|$ maximal. If $\mathcal{T}$ differs from $\mathcal{T}'$, there is an edge $e^* \in L$ such that $L' \cup e^*$ contains a cycle. Now, let S^* be the vertex set in Algorithm 1.1 just before e^* was

added to L. Further, let $i'j' \in L'$ be a second edge on the cycle with $i' \in S^*$, $j' \in V \setminus S^*$. Then, $(V, L' \cup \{e^*\} \setminus \{i'j'\})$ also is a spanning tree, but $\kappa(L' \cup \{e^*\} \setminus \{i'j'\}) \leq \kappa(L')$. Hence, either $\mathcal{T}'$ was not optimal or $|L \cap L'|$ was not maximal.

The time to compute a minimum cost spanning tree with the Dijkstra-Prim algorithm is $\mathcal{O}(|E| + |V|\log|V|)$ by using a data structure such as a *Fibonacci heap* for sorting the potential spanning tree edges. Hence, the problem is in P. Further, the Dijkstra-Prim algorithm is an example of a *greedy algorithm*; cf. Section 1.4.3. Another greedy algorithm for the minimum cost spanning tree problem is *Kruskal's algorithm* [69]. It starts with the trivial forest (V, L), $L = \emptyset$, and repeatedly adds the edge of minimum cost such that the new subgraph (V, L) remains a forest. After adding $|V| - 1$ edges a minimum cost spanning tree is found (which can be proved by a similar argument as above).

Recently, there has been renewed interest in the minimum spanning tree problem. However, the tree is restricted to have a bounded degree at all vertices. In contrast to the minimum cost spanning tree problem, no polynomial-time algorithm is known for the *bounded degree minimum cost spanning tree* problem [10, 51]. NP-hardness follows from the Degree-Constrained Spanning Tree problem, where one has to minimize the maximum degree of the spanning tree [50, Problem ND1].

Minimum Steiner Tree Problem

Application 1.2 *Reconsider the situation of Application 1.1. Instead of a new office building, we consider an existing building with a wired LAN infrastructure in place. The new backbone for the WLAN can use this infrastructure but is not required to do so. A set of connection points at the wired LAN network have been identified, and the costs to connect an AP to a connection point are again denoted by κ_{ij}.*

In this case the set V consists of all APs and all connection point to the wired LAN network. Since only the APs have to be connected to each other, S consists of the APs only and H is complete. The connection cost between any two connection points $i, j \in V \setminus S$ can be defined as $\kappa_{i,j} = 0$. This problem is known as the *minimum cost Steiner tree problem*. The subset $U \subset V$ are the so-called *terminals* that have to be connected, whereas the remaining vertices $V \setminus U$ can be used as *hubs* to save connection costs.

The minimum cost Steiner tree problem cannot be solved in polynomial time, unless P $=$ NP [66]. Even for grid graphs the problem is NP-complete [49]. The special case where $|U| = 2$ requires a path between the two end nodes and is better known as the *shortest path problem*; see Section 1.5.2.1.

Among the many different approaches that have been developed to find optimal, or at least very good, solutions for the Steiner tree problem, polyhedral methods have been particularly successful. The problem can be formulated as an integer linear program in many different ways. We present here one formulation as a warm-up and refer to the survey by Voß [89] for further formulations.

For every $e \in E$ we introduce a binary variable x_e indicating whether ($x_e = 1$) or not ($x_e = 0$) the edge is part of the Steiner tree. The Steiner Tree Problem now reads

$$\min \quad \sum_{e \in E} \kappa_e x_e \tag{1.7a}$$

$$\text{s.t.} \quad \sum_{e \in \delta(W)} x_{ij} \geq 1 \qquad \forall W \subset V, W \cap S \neq \emptyset, (V \setminus W) \cap S \neq \emptyset \tag{1.7b}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{1.7c}$$

The *Connectivity constraints* (1.7b) ensure that there is always at least one edge selected to connect a terminals in W and terminals in $V \setminus W$. This in particular guarantees that all terminals are connected in a feasible solution, and hence the minimum Steiner tree can be found by solving (1.7).

Although this is not the strongest formulation (in the sense of value of the LP relaxation, cf. [27]), the way of formulating the problem is already of help for more complex problems like the two-layer network design problem discussed in Chapter 3, where inequalities (1.7b) with $|W| = 1$ are used as cutting planes (cf. page 108). For further information on the Steiner tree problem we refer to Polzin [80] and Voß [89]. SteinLIB [68] is a library with benchmark instances for the Steiner tree problem.

Minimum Steiner Network Problem

Application 1.3 *An Internet Service Provider (ISP) provides a Virtual Private Network (VPN) service to its corporate customers. A VPN is a computer network that uses virtual circuits in a larger network. For each customer a VPN has to be established that enables communication between all the customer's locations. The ISP leases bandwidth connections at a telecommunication network operator. Costs savings can be achieved by combining the requirements of multiple corporate customers.*

The remaining case generalizes the Steiner tree problem. If not all vertices in U have to be mutually connected, the topology does not necessarily need to be a tree spanning U. For each component of H the optimal topology will be connected, but there is no need to connect vertices in different components. In some cases, however, it might be beneficial to connect vertices in the same component of H via vertices from another component. In fact, the solution will be a forest with at most the same number of components (i.e., trees) as H. For more information on this problem and a primal-dual approximation algorithm, we refer to [6].

1.5.1.2 Design of Ring Topologies

From a reliability point of view, a network that connects the desired vertex pairs with exactly one path is not necessarily a good solution. If a link fails the topol-

ogy is disconnected and no communication is possible anymore between the two components of the network.

To improve the reliability of the network, it has been suggested to use a ring as topology structure instead of a tree. The cheapest way to guarantee that two paths exist between every pair of nodes is the selection of a *Hamiltonian cycle* (i.e., a single cycle connecting all nodes in the network). Not only is this problem already NP-hard to solve (it is equivalent to the traveling salesman problem) but the potential topology might not contain a Hamiltonian cycle. Further, the average path length of the two paths is $\frac{n}{2}$ for a network of n nodes. Accordingly, communication delays will be high. Therefore, an alternative is to design a network composed of connected rings. Many different approaches towards this problem exist; we refer to [60] for an introduction.

1.5.1.3 Design of Meshed Topologies

Instead of connecting the network nodes via a ring, more general configurations can be considered that provide a higher reliability than tree or ring networks. To simplify the presentation, we only consider link failures and undirected graphs in the following. For this purpose, we associate with every edge $\{i, j\}$ of the graph $H = (V, F)$ (as defined in Section 1.5.1.1) a value r_{ij} denoting the minimum *connectivity* between i and j, i.e., the minimum number of edge-disjoint paths. Now, a *meshed network topology* is a subset $L \subseteq E$ such that for all $\{i, j\} \in F$, i and j are r_{ij}-connected. The integer linear programming formulation (1.7) can be easily adapted to this situation by replacing the right-hand side of (1.7b) with $\max_{i \in W, j \in V \setminus W} r_{ij}$. The minimum cost meshed network topology problem can be solved by integer linear programming techniques. Some references are [48, 56–58].

1.5.2 Network Routing Problems

Given a network topology, communication between two nodes can be established according to a set of rules specifying the path the signal should follow. Depending on the technology many different *routing protocols* exist. In this section we discuss some of the optimization problems providing a basis in network routing optimization. At the end, we refer to more elaborate routing problems discussed in the chapters. Network flows have been extensively studied, and [7] provides a rich collection of results in this area.

1.5.2.1 Routing of a Single Commodity

The most elementary problem in network routing optimization is the choice of a routing for a single commodity, e.g., one source-destination pair. This problem can

occur at the planning stage (e.g., simultaneous planning of communication paths) or at the operational stage (e.g., *online call admission*). The problem also has to be solved in many multi-commodity network optimization problems where dynamic column generation is used (cf. Section 1.5.3).

Shortest Path Problem

Application 1.4 *Consider the operation of an optical fiber network. The call admission officer is asked to set up a new 10 Gbit/s connection between two core nodes. To avoid re-sorting of the packets at the receiving end of the connection, a single 10 Gbit/s lightpath has to be used. To limit the packet delay to a minimum, the path should be as short as possible.*

If no further routing restrictions apply, this problem can be described as follows. Given a digraph $D = (V, A)$ (e.g., arc $a \in A$ exists if and only if enough spare capacity exists), a weight function $w : A \mapsto \mathbb{Q}$ (e.g., the delay) and a source-destination pair $s, t \in V$, find a path p from s to t such that $\sum_{a \in p} w_a$ is minimized. In general the weight function w can represent any relevant metric, e.g., length of the network links, delay, spare capacity, etc..

The above problem is well-known as the *shortest path problem*. If all weights are nonnegative, the problem can be solved in polynomial time by a wide variety of algorithms, the most famous one being *Dijkstra's $\mathcal{O}(|V|^2)$ algorithm* [39]; see Algorithm 1.2. An excellent source for faster implementations of Dijkstra's algorithm and other faster algorithms for the shortest path problem is [83, Chapter 7]; see also [7].

Algorithm 1.2 Dijkstra's algorithm to determine the shortest path p between $s \in V$ and $t \in V$ in a weighted digraph $D = (V, A, w)$.

Define $d : V \mapsto \mathbb{Q}_+$ by $d(i) := 0$ if $i \equiv s$ and $d(i) := \infty$ otherwise
Define $p : V \mapsto V$ by $p(i) := i$ for all $i \in V$
Let $S := \emptyset$
while $t \notin S$ **do**
   Let $i = \arg\min\{d(j) \mid j \in V \setminus S\}$
   for $j \in N^+(i)$ **do**
      if $d(i) + w_{ij} < d(j)$ **then**
         $d(j) := d(i) + w_{ij}$
         $p(j) := i$
   $S := S \cup \{i\}$
if $d(t) \equiv \infty$ **then**
   return No path from s to t exists
Let path p be $(i_0, a_1, i_1, \ldots, a_k, i_k)$ such that $i_j = p(i_{j+1})$ and $i_0 = s$, $i_k = t$.
return p and $d(t)$

The function $d : V \mapsto \mathbb{Q}_+$ keeps the shortest distance *known* from s to any vertex, whereas $p(i)$ records the *previous* vertex on the shortest path from s to i. Dijkstra's

algorithm repeatedly selects vertices $i \in V$ for which the distance $d(i)$ to the source vertex s cannot be improved anymore and updates the distance of vertices that can be reached in a single step from this vertex. Since all arc weights are nonnegative, the distance to none of the vertices in S can be decreased this way, and hence as soon as $t \in S$, the distance $d(t)$ is final.

In case some arc weights are negative, the above algorithm cannot guarantee an optimal solution anymore (i.e., if negatively weighted cycles exist) and the problem can be proved to be NP-complete [50, 66]. This result is of particular importance for optimization approaches that exploit dynamic column generation since in particular cases negative arc weights might appear. Shortest path algorithms are used in many situations; see, e.g., Chapters 4, 5, and 8.

Suurballe's Problem

Application 1.5 *Reconsider the situation of Application 1.4. To guarantee high availability of the connection, two vertex-disjoint paths with minimum total delay have to be installed. All packets are transmitted by both lightpaths. At the receiving end, the packets that arrive earliest are processed, whereas the other ones (if arriving) are discarded.*

This closely related and widely studied problem in communication networks is known as *Suurballe's problem* [86], often wrongly referred to as *Suurballe's algorithm*. Instead of a shortest path between s and t, a *shortest cycle* containing s and t is searched for. Stated differently, and more generally, we have to find K paths $p_1, \ldots, p_K$ between s and t such that $\sum_{j=1}^{K} \sum_{a \in p_j} w_a$ is minimized. The paths must be either arc-disjoint or vertex-disjoint (except for source and destination), depending on the setting. For $K = 2$ and vertex-disjoint paths, the problem reduces to the shortest cycle problem.

This problem occurs in the context of *1+1 dedicated path protection*, where a *working* or *primary path* and a disjoint *backup path* have to be selected. From a practical perspective there might be several reasons to balance the length of both paths, e.g., comparable delays, or avoidance of a capacity imbalance between primary and backup capacity. In such situations we might apply a *successive shortest path* computation; i.e., we first compute the shortest path between s and t and next we compute a shortest path in the augmented digraph where all arcs (and nodes) on the shortest path are removed. However, such a second path might not exist, although there exists a shortest cycle; see Figure 1.9.

Specific algorithms have been developed for this problem, in particular using Dijkstra's algorithm; cf. [17]. Alternatively, the problem can be solved as a *minimum cost flow problem*.
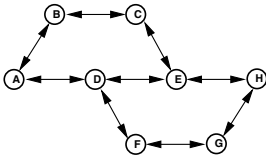
Fig. 1.9 A digraph with arc weights $w_a = 1$ for all $a \in A$ where the successive shortest path algorithm fails to find two vertex-disjoint paths between vertices A and H

Minimum Cost Flow Problem

Application 1.6 *A world-wide operating investment bank needs access to 250 Gbit/s of bandwidth between its European and US headquarters. For this it can lease bandwidth from a number of network operators for sections of the connection, represented by a directed graph. Bandwidth on a section can be leased with a modularity of 10 Gbit/s and is priced accordingly.*

The minimum cost flow problem is defined by a digraph $D = (V, A)$, a cost function $w : A \mapsto \mathbb{Q}_+$, a capacity function $c : A \mapsto \mathbb{Z}_+$, and a supply/demand function $d : V \mapsto \mathbb{Z}$. A vertex $i \in V$ is called a *supply vertex* if $d_i > 0$, a *demand vertex* if $d_i < 0$, and a *transit vertex* if $d_i = 0$. A *flow* $f : A \mapsto \mathbb{Q}_+$ is an assignment of values to the arcs of the digraph. We call a flow *proper with respect to d* if the *flow conservation* constraints

$$\sum_{a \in \delta^+(i)} f_a - \sum_{a \in \delta^-(i)} f_a = d_i \tag{1.8}$$

hold for all $i \in V$. We further call a flow *proper with respect to c* if the *capacity constraints*

$$f_a \leq c_a \tag{1.9}$$

hold for all $a \in A$. A *minimum cost flow* is a proper flow with respect to d and c that minimizes

$$\sum_{a \in A} w_a f_a .$$

The description of the minimum cost flow problem is more general than Application 1.6 requires: there is only one supply vertex and only one demand vertex. Suurballe's problem with arc-disjoint paths can be easily formulated as a minimum cost flow problem by setting $c_a = 1$ for all $a \in A$, and

$$d_i = \begin{cases} K & \text{if } i = s, \\ -K & \text{if } i = t, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Every proper flow with respect to d and c can now be translated to a set of K arc-disjoint paths. For each path p_j we greedily select successive arcs for which $f_a = 1$, starting at the source s and ending at the destination t, setting $f_a = 0$ as soon as it is selected.

Because of the *total unimodularity* of the constraint matrix in the minimum cost flow problem, the solution is integer valued if all arc capacities c_a are integral. This property is, for example, relevant in the context of optical networks where we would like to route a number of lightpaths between a source vertex $s \in V$ and a target vertex $t \in V$.

Maximum Flow Problem

Another problem closely related to the above problems is the *maximum flow* problem. Here the input consists of a digraph $D = (V,A)$, a capacity function $c : A \mapsto \mathbb{Z}_+$, and two designated vertices, a source $s \in V$ and a target $t \in V$. The objective is to maximize the flow between s and t.

This problem solves the feasibility version of Application 1.6: can we allocate 250 Gbit/s of bandwidth between the two headquarters? It also can be used to test whether K arc-disjoint paths between s and t exist, as requested by Suurballe's problem.

The problem is in particular known for the famous *max-flow min-cut theorem*. A *cut* C in a digraph $D = (V,A)$ is a subset of the arcs such that $(V,A \setminus C)$ is disconnected. A cut is called an $s - t$ cut if s and t are not connected anymore.

Theorem 1.7 (Dantzig and Fulkerson [37]). *Let $D = (V,A)$ be a digraph with source node $s \in V$ and target node $t \in V$. Let $c : A \mapsto \mathbb{Z}_+$. Then the maximum value of an $s - t$ proper flow with respect to c is equal to the minimum capacity of an $s - t$ cut.*

Combinatorial algorithms to solve the minimum cost network flow problem or the maximum flow problem can be found in [7, 32] or any good textbook in combinatorial optimization.

1.5.2.2 Routing of Multiple Commodities

In general, a *commodity* is a good that does not require further differentiation, either from a practical point of view or from a mathematical point of view. The minimum cost flow problem of the previous section is a good example of a single commodity transportation problem. There may be multiple supply vertices and multiple demand vertices, but there is no differentiation between the goods supplied and demanded. Supply from any vertex can be used to fulfill the demand.

Multi-commodity Flow Problem

In communication networks, it is usually of highest importance that *goods* be sent between the right network nodes. If more than one pair of network nodes is considered simultaneously, a *multi-commodity flow* problem has to be solved from a technological point of view.

Multi-commodity Edge-Flow Formulation

Application 1.8 *Let $D = (V, A)$ represent a network and the function $c : A \mapsto \mathbb{Z}_+$ denote the available bandwidth. Let a set of point-to-point demands be represented by another weighted digraph $H = (V, B)$ with weight function $d : B \mapsto \mathbb{R}_+$ (d_{ij} denotes the required bandwidth between i and j, $ij \in B$). Find a routing of the demands such that the total spare capacity is maximized.*

Depending on the real application, the routing might have to satisfy additional requirements; see Section 1.5.2.3. For now, we assume that the routing can be *bifurcated*, i.e., it can be split among different paths from source to destination. Accordingly the problem can be modeled as a linear program. For each $st \in B$, we introduce a set of edge-flow variables f_a^{st} to model the flow on arc $a \in A$ between source i and target j. The Multi-commodity Flow (MCF) problem now reads

$$\min \quad \sum_{st \in B} \sum_{a \in A} f_a^{st} \tag{1.10a}$$

$$\text{s.t.} \quad \sum_{a \in \delta_D^+(i)} f_a^{st} - \sum_{a \in \delta_D^-(i)} f_a^{st} = \begin{cases} d_{st} & \text{if } i = s \\ -d_{st} & \text{if } i = t \\ 0 & \text{otherwise} \end{cases} \quad i \in V, st \in B \tag{1.10b}$$

$$\sum_{st \in B} f_a^{st} \le c_a \qquad\qquad\qquad a \in A \tag{1.10c}$$

$$f_a^{st} \ge 0 \tag{1.10d}$$

Instead of maximizing the total spare capacity, the objective (1.10a) models the minimization of the used resources. With fixed capacities c_a these objectives are equivalent. Constraints (1.10b) model the flow conservation from source to target for all demands, whereas (1.10c) model the capacity constraints. Compared to the single commodity case, the latter take all commodities simultaneously into account, whereas for the flow conservation a separate set of constraints is set up for each commodity.

The MCF (1.10) is a linear program and therefore can be solved efficiently; cf. Section 1.3. However, for larger networks the size of the MCF model is a reason for concern. Assuming a demand between every node pair (i.e., H is complete), the model has $\mathcal{O}(n^2 m)$ variables and $\mathcal{O}(n^3)$ constraints, where $n = |V|$ and $m = |A|$.

From a mathematical point of view there is no need to differentiate between demands as long as they have either the same source or the same target. In this

way, the size of (1.10) can be reduced to $\mathcal{O}(nm)$ variables and $\mathcal{O}(n^2)$ constraints by aggregation of demands into n commodities (instead of $\mathcal{O}(n^2)$). We construct a set K of commodities from the point-to-point demands as follows. For every source node s, we define a commodity $k \in K$ by considering all demands represented by $\delta_H^+(s)$. The demand value

$$d^k = \sum_{t \in N_H^+(s)} d_{st}$$

of the commodity is simply defined as the total demand value of its demands. For any node $i \in V$, let

$$d_i^k := \begin{cases} d^k & \text{if } i = s, \\ -d_{si} & \text{if } i \neq s \end{cases}$$

be the supply (positive) or demand (negative) for commodity k at node i. By construction, the equality $\sum_{i \in V} d_i^k = 0$ holds for every commodity k. Now MCF can be reformulated with continuous aggregated flow variables f_a^k:

$$\min \quad \sum_{k \in K} \sum_{a \in A} f_a^k \tag{1.11a}$$

$$\text{s.t.} \quad \sum_{a \in \delta_D^+(i)} f_a^k - \sum_{a \in \delta_D^-(i)} f_a^k = d_i^k \qquad i \in V, k \in K \tag{1.11b}$$

$$\sum_{k \in K} f_a^k \leq c_a \qquad a \in A \tag{1.11c}$$

$$f_a^k \geq 0 \tag{1.11d}$$

Given a solution $(f_a^k)_{k \in K, a \in A}$, the routing of every point-to-point demand can be reconstructed by greedily selecting a path between source s and target t of a point-to-point demand, setting its value to the minimum of the demand value d_{st} and the minimum flow value f_a^k on the arcs along the path, and reducing the flow values f_a^k on the path with this value.

Multi-commodity Path-Flow Formulation

Alternatively, one can formulate the MCF problem by variables representing the paths between source and target. For this, let $\mathcal{P}^{st}$ denote all paths in D between s and t and let $\mathcal{P} = \cup_{st \in B} \mathcal{P}^{st}$ (note that there can be exponentially many paths between two vertices). Further, let $\mathcal{P}_a = \{p \in \mathcal{P} : a \in p\}$. For every $st \in B$ and $p \in \mathcal{P}_{st}$ we define a *path-flow* variable y_p^{st} denoting the flow using this path. The MCF now reads

$$\min \quad \sum_{st \in B} \sum_{p \in \mathscr{P}^{st}} y_p^{st} \tag{1.12a}$$

$$\text{s.t.} \quad \sum_{p \in \mathscr{P}^{st}} y_p^{st} = d_{st} \qquad\qquad st \in B \tag{1.12b}$$

$$\sum_{st \in B} \sum_{p \in \mathscr{P}_a} y_p^{st} \le c_a \qquad\qquad a \in A \tag{1.12c}$$

$$y_p^{st} \ge 0 \tag{1.12d}$$

Although this formulation has an exponential number of variables, it also has its benefits. First of all, since many variables will be 0 in an optimal solution, not all variables have to be considered explicitly by exploiting *dynamic column generation*; cf Section 1.4.1. Given the explicit consideration of the variables y_p^{st} for a subset $\mathscr{P}' \subset \mathscr{P}$ of the paths, (1.12) is solved to optimality if and only if for all $st \in B$ there does not exist a path $p \in \mathscr{P}^{st} \setminus \mathscr{P}'$ with

$$\sum_{a \in p} \mu_a < 1 - \pi^{st}, \tag{1.13}$$

where μ_a and π^{st} are the *dual variables* corresponding to (1.12c) and (1.12b), respectively. To test whether there exists such a path p, we have to solve a shortest path problem on $D = (V, A)$ with weights $\pi^a \ge 0$ for all $a \in A$. If the length of the shortest path strictly smaller than $1 - \pi^{st}$, the variable has to be considered explicitly.

Another benefit of the path-flow formulation (1.12) is that restrictions on the routing paths can be taken into account, e.g., paths with less than K hops (number of arcs) or with delay below a certain threshold. If the number of paths that satisfy such requirements is still large and dynamic column generation is deployed, the pricing problem becomes a *shortest weight-constrained path problem* which is NP-complete in general [50].

1.5.2.3 More Network Routing Problems

The MCF problem forms the basis for many more complex network routing problems occurring in practice. While capacity planning problems are presented in Section 1.5.3, the remainder of this section contains an (author-biased) sample of routing problem variations.

Multi-commodity Flow in Undirected Graphs

Depending on the technology, the MCF problem might be defined for undirected graphs $G = (V, E)$ and $H = (V, F)$ (note that source and target of a demand are chosen arbitrarily in this case). Formulation (1.12) can be easily adapted to this case by replacing D with G and by redefinition of $\mathscr{P}^{st}$ to be all undirected paths in G between s and t. For the other formulations, we have to define a directed graph

$D(G) = (V,A)$ with $A = \{(i,j),(j,i) \mid \{i,j\} \in E\}$. Instead of constraints (1.10c), the following inequality has to be satisfied:

$$\sum_{st \in F} \left(f^{st}_{(i,j)} + f^{st}_{(j,i)} \right) \leq c_{\{i,j\}} \qquad\qquad \{i,j\} \in E \qquad\qquad (1.14)$$

Integer Flow

In the *integer flow problem* we have to route every demand in integral units from source to target. A natural prerequisite is that the demand values d_{st} be integral as well. This problem has to be solved in the context of optical networks [93, 94]. The demand graph represents the number of lightpaths needed between the incident vertices. The formulations (1.10)–(1.12) can be easily adapted to this situation by changing the domain of the variables to the nonnegative integers.

In the single-commodity case, this problem can be solved in polynomial time as a minimum cost flow problem; see Section 1.5.2.1 (fractional capacities can be rounded down without loss of generality). For the multi-commodity case, the problem is in general NP-complete since it contains the Disjoint Connecting Paths problem as a special case [50]. In the Disjoint Connecting Paths problem one has to find k mutually vertex-disjoint paths in $D = (V,A)$ connecting the disjoint vertex pairs $(s_1,t_1),\ldots,(s_k,t_k)$. By graph transformation and setting $c_a = 1$ for all $a \in A$ we obtain an instance of the integer flow problem, and hence NP-completeness of the integer flow problem is shown.

Unsplittable Flow

In the *unsplittable flow* or *non-bifurcated flow problem* we have to route every demand on a single path between source and target. This problem can be modeled by replacing the variables f^{st}_a in (1.10) by $d_{st}x^{st}_a$, where x^{st}_a are binary variables denoting the usage of a certain arc. The flow conservation constraints (1.10b) can be simplified by dividing all coefficients by d_{st}.

In the single commodity case, this problem is simply a shortest path problem where all arcs with $c_a < d_{st}$ are removed from the digraph. The multi-commodity case is NP-complete by the same reduction as that for the integer flow problem.

If continuous instead of binary variables are used the problem is equivalent to the multi-commodity flow problem. However, the variables describe the percentage of flow routing along an arc, which might be beneficial in certain circumstances, e.g., with multiple demand graphs in the case of multi-hour routing.

Routing of Multicasts

Usually this problem is considered in undirected graphs and no distinction between source and targets is made. In the *multicast routing problem* every commodity $k \in K$ consists of a set of terminals T^k and a demand value d^k. A solution consists of a *spanner* for every commodity $k \in K$ connecting all terminals T^k, i.e., a subset of the edges forming a connected subgraph containing all terminals. This problem is therefore known as the *spanner packing problem*, or in the case where the spanners have to be trees, the *Steiner tree packing problem*. A study on these problems can be found in [18]. Multicasting scenarios are also studied in Chapters 2 and 7.

1.5.3 Network Planning Problems

In many networking problems, the link capacities are not fixed but can be chosen at certain costs. In such cases, a *network planning problem* (also called *network design problem* or *network loading problem*) has to be solved. Rich literature exists discussing solution approaches for a wide variety of these problems, e.g., unsplittable, integral, and splittable flows, directed and undirected networks, and directed and undirected demand graphs. A detailed discussion can be found in [79]. Here, we restrict ourselves to splittable flows in a directed network and directed demand graph. We discuss two different approaches before a third approach is described in more detail for the case of optical networks (Section 1.5.4).

1.5.3.1 Linear Cost Functions

Application 1.9 *A regional service provider leases bandwidth from different network operators active in his area. Each of the network operators offers bandwidth at the links of a network $D = (V, A)$. For each of the links $a \in A$, κ_a denotes the rate at which the operator cheapest for this link is offering bandwidth. Let $H = (V, B)$ represent the point-to-point demands with demand function $d : B \mapsto \mathbb{R}_+$. Find a routing of the demands such that the leasing cost is minimized.*

Compared to the multi-commodity flow problem of Section 1.5.2.2, this problem has a different objective and different capacity constraints. The flow conservation constraints (1.10b) remain the same. The new objective reads

$$\min \quad \sum_{a \in A} \kappa_a z_a \tag{1.15a}$$

where z_a is nonnegative variable denoting the bandwidth consumption of arc $a \in A$. The capacity constraint 1.10c is replaced by

$$\sum_{st \in B} f_a^{st} \leq z_a \qquad\qquad a \in A \qquad\qquad (1.15b)$$

Assuming $d_{st} \geq 0$, the nonnegativity constraints (1.10d) guarantee that the values of z_a will also be nonnegative.

So far, this problem is not really exciting. Not only can it be solved in polynomial time (as it is a linear program), in any optimal solution we will have $z_a \equiv \sum_{st \in B} f_a^{st}$. Moreover, the point-to-point demands can be routed completely independently without losing optimality. Hence, the problem consists of $|B|$ shortest path problems, where κ_a define the lengths of the arcs. Furthermore, the result will be an unsplittable flow and thus the problem with unsplittable flow is also solved to optimality.

The problem becomes a different game if shared protection mechanisms such as Single Backup Path Protection (SBPP) are exploited. In SBPP the demand not only has to be routed along a single working path, but a backup path has also to be specified and is used if one of the links on the working path fails. By assuming that at most one link can fail at a time, several backup paths can share bandwidth capacity (i.e., those for which the working paths are disjoint). This problem is already NP-hard [85] with continuous bandwidth capacity variables. Survivable networks are studied in more detail in Chapter 5.

1.5.3.2 Discrete Cost Functions

Let us now turn to the case where capacity can only be installed in certain amounts. The problem instantly becomes more difficult. If capacities can be installed in only one size, the model remains the same by scaling the installed bandwidth to 1. Only the integrality constraints of the z_a variables have to added.

More generally, different discrete capacities $C^1 < C^2 < \ldots < C^M$ can be installed against cost $\kappa_a^1 < \kappa_a^2 < \ldots < \kappa_a^M$. In such a case, we introduce capacity variables $z_a^1, \ldots, z_a^M \in \mathbb{Z}_+$ to denote the number of times a particular bandwidth *module* is installed on a particular arc. The objective now changes to

$$\min \quad \sum_{a \in A} \sum_{m=1}^{M} \kappa_a^m z_a^m \qquad\qquad\qquad (1.16a)$$

whereas the bandwidth capacity constraint (1.15b) now reads

$$\sum_{st \in B} f_a^{st} \leq \sum_{m=1}^{M} C^m z_a^m \qquad\qquad a \in A \qquad\qquad (1.16b)$$

This problem is NP-hard for many special cases [19, 26]; it is, however, relevant for a variety of technologies; see [90] for an example. Integer linear programming has been used intensively to solve these problems (see [90] for details and references); an alternative approach is discussed in Section 1.5.4.

1.5.3.3 More Network Planning Problems

As already pointed out, the discussed planning problems are at the core of more complex network design problems. In-depth discussions of these can be found in books such as [59, 79]. Two particular cases are discussed in this book. Chapter 3 discusses a multilayer network design problem whereas Chapter 8 is devoted to shortest path network routing and planning; see also Section 1.6.1. A good resource for realistic network planning instances is the SNDlib website [77].

1.5.4 A Randomized Cost Smoothing Approach for Optical Network Design

Contributed by Alpár Jüttner[1]

In info-communications network design, the cost of optical ports and links grows in discrete steps as the capacity is being increased. This cost function is referred to as "step function" or "staged capacity cost."

In this section, we propose and compare methods that perform randomized smoothing of these staged capacity cost functions to allow decomposition of the network design problem into a sequence of weighted shortest path searches.

1.5.4.1 Introduction

During an optical network design, not only the topology but also the demand routing and link/port capacities have to be determined. For example, in an SDH/SONET network the capacities, i.e., the interface speeds, take values of 155.52; 622.08; 2,488.32; 9,953.28 and 39,813.12 Mbit/s, i.e., they always multiply by exactly four. In optical transport networks (OTNs) the capacities take values of 2,666, 10,709, and 43,018, i.e., values always multiply by a bit more than 4 (4.017). Furthermore, in OTN and in any other Coarse WDM (CWDM) or Dense WDM (DWDM) system one or more wavelengths can be used in parallel for demands of larger capacity, and the number of wavelengths to be used in a WDM system varies in steps of 8, 16, 24, 32, 40, 48, 64, 80, 96, 120, and so on, wavelengths per fiber. This shows that for all-optical networks we face staged capacity costs.

This section proposes a simple yet efficient approximation approach to handle this problem. The methods presented here can be used for green-field design, network extension, or configuration purposes (e.g., VPN, leased λ, and leased line services), as will be formulated in Section 1.5.4.2.

---

[1] Alpár Jüttner

Department of Operations Research, Eötvös University, Pázmány P. s. 1/C, H-1117 Budapest, Hungary, e-mail: `alpar@cs.elte.hu`

1.5.4.2 General Network Planning Problem (GNPP)

The network is represented as a directed graph $G_{net} = (N, L)$ with the set N of nodes and the set L of the possible links. The demands are also represented by a directed graph $G_{dem} = (N, D)$ over the same node set N and a function $dem : D \mapsto \mathbb{R}_+$. Each demand is represented by an edge $d \in D$, where the tail and the head of d show the source and the target of the demands, respectively, while $dem(d)$ is the amount of the demand. Moreover, we are given a monotone increasing load-dependent cost function on each link, denoted by $cost : L \times \mathbb{R}_+ \mapsto \mathbb{R}_+$, so the cost of establishing a link ℓ with capacity c (or upgrading ℓ to capacity c) is $cost(\ell, c)$.

Then, the task is to assign a route p_d to each demand d in such a way that the obtained configuration minimizes the total cost

$$\sum_{\ell \in L} cost(\ell, traffic(\ell)), \quad \text{where} \quad traffic(\ell) = \sum_{d:\ell \in p_d} dem(d). \tag{1.17}$$

This definition models several usual network optimization problems. Some examples are shown below.

Routing Configuration

Let us assume that we are given a network with given link capacities $cap(\ell)$ and the set of demands we want to carry over this network while keeping the capacity constraints or minimizing the total overload. This problem can be modeled by the framework above by defining the following cost function.

$$cost(\ell, t) = \max(0, t - cap(\ell)) \tag{1.18}$$

If we want to minimize the number of overloaded links instead of the total overload, we can use the following

$$cost(\ell, t) = \begin{cases} 0, & \text{if } t \leq cap(\ell) \\ 0, & \text{if } t > cap(\ell) \end{cases} \tag{1.19}$$

Network Design

This case is a green-field network planning problem: we want to plan a new network carrying our traffic and we want to minimize the installation costs. Then L will consist of all possible links, while $cost(\ell, t)$ is defined to be the cost of installing a link of capacity t between the source and the destination of ℓ. As we can choose only from some fixed capacity equipment, the cost will be a step function for each possible link ℓ.

Network Extension

This case is similar to the previous one, but now we have some existing links, which can be used with no extra installation cost. In this scenario, the cost $cost(\ell,t)$ of an existing link ℓ with capacity $cap(\ell)$ is 0 for all $t \leq cap(\ell)$.

1.5.4.3 Solution Methods

Local Search

The following simple observation plays a central role in solving GNPP.

Claim. Let us assume that we have fixed a route for all but one demand, i.e., we have fixed p_d for all $d \in D \setminus d'$. Then the optimal route for the last demand d' can be found by searching for a shortest path (using Dijkstra's algorithm) according to the following auxiliary length function.

$$len(\ell) = cost(\ell, traffic_{d'}(\ell) + dem(d')) - cost(\ell, traffic_{d'}(\ell)), \tag{1.20}$$

where

$$traffic_{d'}(\ell) = \sum_{d:d \neq d' \wedge \ell \in p_d} dem(d). \tag{1.21}$$

Using the claim above, a simple heuristic can be given as follows. We start with an arbitrary solution; then in each iteration we remove a route from the configuration and replace it by the locally best alternative. We repeat this process until no more improving change is possible. We refer this as the *Local Search (LS) method*.

It is easy to see that this method finds the theoretically optimal solution if the cost function is linear in the second variable. However, it works poorly for the two most typical cost functions, the concave and the staged costs, especially for the latter one.

Cost Function Smoothing (CFS) Algorithm

The main weakness of the LS scheme is that it considers the cost function as a black box and queries its values only for the current traffic on the links. So, it does not take into consideration how much traffic can be still allocated on the link before we actually reach the next stage in the cost function, or how much should be deallocated for realizing real cost decrement.

A solution proposed by [71] is to smooth the actual cost function by convolving it with a "Gauss-like" function. This smoothed cost function $cost_\delta(\ell,t)$ should be parameterized in such a way that for large δ values it provides a very smooth function (close to linear for practical t values) and it approaches $cost(\ell,t)$ as δ approaches 0 (see Figure 1.10).

Then, the CFS algorithm is the same as LS with the exception that we use $cost_\delta(\ell,t)$ instead of the original cost function. We start with a large δ value and decrease it exponentially during the execution of the algorithm.
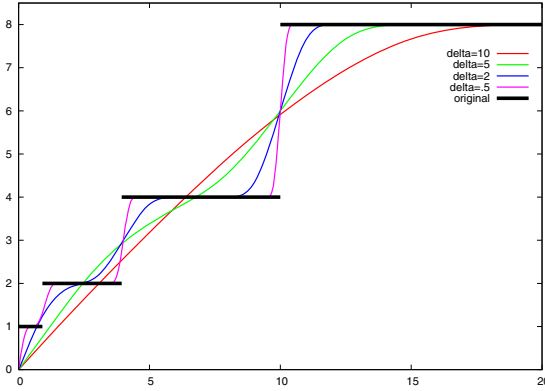
Fig. 1.10 $cost_\delta(\ell,t)$ with different δ values

A concrete example for such a smoothing is the following (this is a refined version of what is given in [71]).

$$cost_\delta(\ell,t) = (cost'(\ell,\cdot) * h_\delta(\cdot))(t) = \int_{-\infty}^{\infty} cost'(\ell,\xi)h_\delta(t-\xi)d\xi, \tag{1.22}$$

where

$$cost'(\ell,t) = \begin{cases} cost(\ell,t) & \text{if } t \geq 0, \\ -cost(\ell,-t) & \text{otherwise,} \end{cases} \tag{1.23}$$

$$h_\delta(t) = \frac{1}{\delta}h\left(\frac{t}{\delta}\right), \tag{1.24}$$

and

$$h(t) = \begin{cases} 1 - 2t^2 & \text{for } |t| \leq 1/2 \\ 2(|t|-1)^2 & \text{for } 1/2 \leq |t| \leq 1 \\ 0 & \text{for } 1 \leq |t|. \end{cases} \tag{1.25}$$

The symmetric definition of the cost function in (1.23) ensures that $cost_\delta(\ell,0) = 0$ holds for all δ. The advantage of smoothing function (1.25) is that the convolution integral is a polynomial for constant or linear segments of the cost functions; thus it is fast to evaluate for staged or piecewise linear cost functions.

Randomized Cost Smoothing (RCFS) Algorithm

This approach is somewhat similar to the Cost Function smoothing, but instead of modifying the cost function, we apply a random "uncertainty" when querying its values. Namely, we change Equation (1.20) to the following.

$$len(\ell) = cost(\ell, traffic_{d'}(\ell) + dem(d') + R_\delta) - cost(\ell, traffic_{d'}(\ell)), \qquad (1.26)$$

where R_δ is a certain random variable of standard deviation δ. Natural choices for R_δ include the Gaussian distribution $N(0, \delta^2)$ and the exponential distribution. If R_δ can also take negative values, then $len'(\ell) = \max(0, len(\ell))$ should be used in order to avoid negative lengths. Note that Dijkstra's algorithm reads the length of each link only once, so this gives a consistent length function in each iteration.
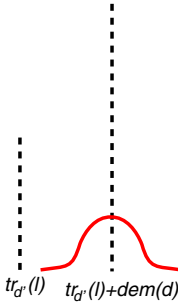
Fig. 1.11 Randomized smoothing of $cost(\ell, t)$

Note that the *expected value* of $cost(\ell, t + R_\delta)$ is the same as $cost(\ell, t)$ smoothed or convolved by the probabilistic distribution function of the random variable R_δ; thus this approach indeed performs a "randomized smoothing." On the other hand, an expected advantage of this scheme is that similarly to other metaheuristics such as Simulated Annealing or the Evolutionary Algorithms, the randomness may provide higher freedom for the algorithms when choosing replacement of a route; thus it may have a higher chance of avoiding the local optima that are far from the global optimum.

1.5.4.4 Evaluation

Here, we demonstrate the behavior of the algorithms on a single but representative network topology. The test network (see Figure 1.12) is a two-connected planar graph consisting of 50 nodes and 84 bidirectional links. It was generated by `lgfgen`, a random graph generator of LEMON [70].

The cost function is a step function, and it is also linearly proportional to the physical length of the link, i.e,
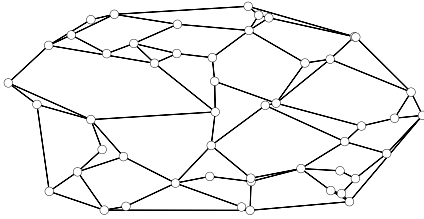
Fig. 1.12 Test network example with 50 nodes and 84 links

Table 1.1 Cost obtained by the different algorithms

	Homogeneous demands	Big trunks
SHORTEST	91101	89575
LS	83371	72601
CFS	59890	56158
RCFS/Gauss	86722	69444
RCFS/Exp	63342	52104

$$cost(\ell,t) = length(\ell)step\_fn(t), \text{ where } step\_fn(t) = \begin{cases} 0 & \text{for } t = 0, \\ 1 & \text{for } 0 < t \leq 1, \\ 2 & \text{for } 1 < t \leq 4, \\ 4 & \text{for } 4 < t \leq 10, \\ 8 & \text{for } 10 < t \leq 100, \\ M & \text{for } 100 < t. \end{cases}$$

Two different types of demand matrices were chosen for the tests.

- Homogeneous demands. In this scenario, we have a homogeneous traffic matrix, i.e., we have the same amount of traffic between any pair of nodes.
- Big trunks. In this scenario only 245 (out of the 2450 possible) random pairs of nodes was chosen as traffic sources and destinations but the demands are 10 times larger than in the previous case.

We ran LS, CFS, and RCFS methods on these examples, and we also computed the cost of the simple shortest path routing (SHORTEST) as a reference. In the case of RCFS, both Gaussian and exponential randomization have been tested. The costs of the obtained solutions are presented in Table 1.1.

The algorithms were implemented in C++, heavily based on the LEMON library [70]. The tests were made on a laptop equipped with a 2 GHz Centrino Duo processor and running the Linux (OpenSuse 10.2) operating system.

The running times of SHORTEST (less than 0.1 second) and LS (a few seconds) are obviously much less than those of CFS and RCFS. Actually, the running times of these algorithms depend on the number of their iterations. Thus, the number of iterations was chosen in such a way that the resulting running time was around 40 seconds. The other parameters were tuned to provide the best results.

It is worth mentioning that RCFS completes around 2,000,000 iterations within 40 seconds while CFS completes only around 400,000 within the same time. This is because the cost function calculation must be performed at each iteration of the algorithms; thus, it turns out to be the most resource-consuming part of the algorithms. Therefore, the easier calculation of the randomized cost function makes the running time of an iteration of RCFS much smaller than that of CFS.

Considering the two versions of RCFS, the results show that the Gaussian version performed quite poorly. In fact, in the case of homogeneous demands, this version is even outperformed by the simple LS method. The possible explanation of the better performance of the exponential-distribution-based version is that this cost randomization can be interpreted as a stochastic prediction of the amount of traffic that will be allocated on the link in the subsequent iterations.

Comparing CFS and RCFS, we obtained that for a homogeneous traffic matrix the winner is CFS, while for uneven traffic with big trunks RCFS outperforms CFS. The reason for this is that for a homogeneous traffic matrix, the traffic can be almost continuously distributed on the links and the cost function smoothing seems to perform really well. On the other hand, if there are fewer but larger unsplittable traffic flows, the problem becomes a combinatorial packing problem, and in this case the RCFS method is able to scan a larger portion of the search space.

1.5.5 Wireless Networking

In wireless networking, two classical combinatorial optimization problems have received a lot of attention. We first discuss the frequency assignment problem in wireless networks which is closely related to the vertex coloring problem. Second, the maximum coverage problem is considered, a problem whose basic version can be modeled as a set covering problem.

1.5.5.1 Assignment of Frequencies

The *frequency assignment problem* (FAP), or *channel assignment problem*, plays an important role in all wireless networks that use the frequency division multiple access (FDMA) technology, the most prominent example being the second generation of cellular networks based on the GSM technology. Other applications include satellite communication, TV broadcasting, military wireless networks, WLANs, and Orthogonal Frequency Division Multiplexing (OFDM) in future wireless networks.

Due to this wide variety of applications, there does not exist a single frequency assignment problem, but many variations. The problem was first mentioned in the 1960s [75] when individual frequencies in the radio spectrum were licensed (and charged) by the authorities, and operators of the first cellular phone networks could financially benefit from intelligent frequency planning. Later, frequencies were licensed in blocks and the objective of the operators changed to minimize the dif-

ference between the highest and lowest frequency to be used. In the 1990s most frequencies were licensed and the popularity of mobile communication forced operators to increase the number of antennae, significantly causing high interference levels whenever the frequency planning was not done carefully. Accordingly the objective changed to the minimization of interference or the maximization of interference-free operated antennae.

All these different objectives have been addressed in a vast number of scientific publications. For an overview on the state of the art of frequency assignment we refer to Aardal et al. [3] and the FAP website [44]. In this section, we will restrict ourselves to the basic modeling of the frequency assignment problem and its relation with the vertex coloring problem in undirected graphs.

Frequency Assignment and Vertex Coloring

All variations of frequency assignment have two features in common.

1. A set of wireless transmitters must be assigned frequencies. For each transmitter there is a set of available frequencies given.
2. The frequencies assigned to two transmitters may incur *interference* of one another, resulting in quality loss of the signal. Two conditions must be fulfilled in order to have interference of two signals:

   - The two frequencies must be close on the electromagnetic band. Harmonics may also interfere due to the Doppler effect, but the parts of the electromagnetic band that are generally selected prevent this type of interference.
   - Connections must be geographically close to each other, so that interfering signals are powerful enough to disturb the quality of a signal.

   Usually, data on the level of interference is provided for every quadruple of two transmitters and two frequencies.

This rather general description is complemented by an objective to be optimized. The problem can be modeled as a graph-theoretical problem by defining an undirected *interference graph* $G = (V, E)$ describing the interference relations. To simplify the presentation we assume in this section that no transmitters are colocated, as is usually the case in cellular phone networks (the description can be easily adapted to such situations; see, for example, [41]). The set V now represents all transmitters. The planner can choose a threshold value representing an acceptable level of interference. An edge $\{i, j\} \in E$ exists if and only if there exists a frequency pair (f, g) such that the interference level for the quadruple (i, f, j, g) exceeds the threshold.

In the most simplified case we only consider so-called *co-channel interference*, i.e., an edge exists if and only if the interference level exceeds the threshold for assigning the same frequency to both transmitters. We further assume that the set of frequencies is equal for all transmitters and the objective is to minimize the number of frequencies under the condition that all interference levels remain below the threshold. In this case, FAP reduces to the well-known *vertex coloring problem*. In

the vertex coloring problem, we have to color the vertices of a graph $G = (V, E)$ such that no two adjacent vertices get the same color. The *chromatic number* $\chi(G)$ is the minimum number of colors needed in a feasible coloring of G. It is not difficult to see that in the above-described case the minimum number of frequencies needed equals $\chi(G)$.

From an algorithmic point of view, this problem can be either treated as a decision problem or an optimization problem. If the number of frequencies is fixed, say K, the planner has to answer the question: "Does there exist a solution without unacceptable interference using at most K frequencies?" which is equivalent to solving the K-COLORABILITY decision problem. This problem is not only NP-hard [66] but also hard to approximate [14]. In the case where the number of frequencies is not yet fixed but must be determined, the optimization version of K-COLORABILITY has to be solved.
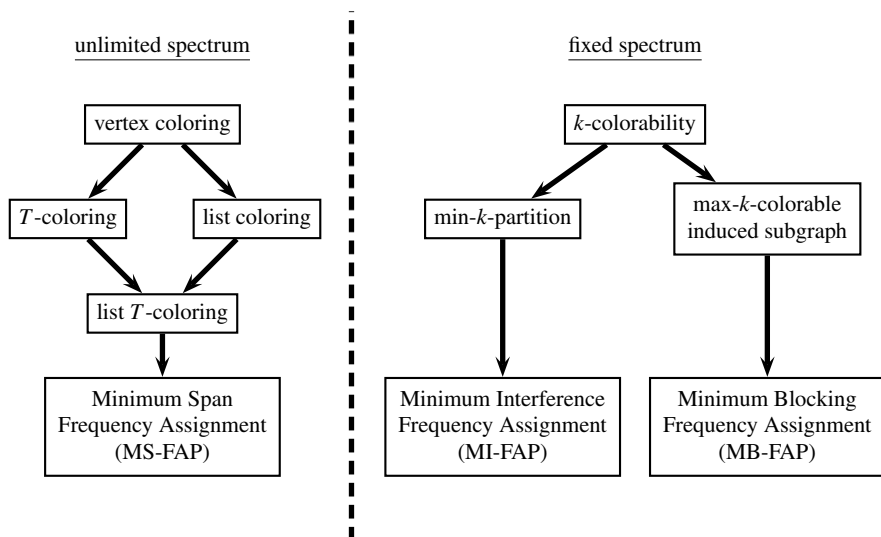
Fig. 1.13 Classification of vertex coloring and frequency assignment problems

This differentiation between choice of frequencies (unlimited spectrum) and fixed frequencies (predetermined spectrum) also holds for more realistic frequency assignment problems; see Figure 1.13. If we include the sets of available frequencies to the vertex coloring problem, a *list coloring problem* has to be solved, whereas the inclusion of interference levels between nonidentical frequencies results in a *T-coloring* problem. Both these variants on the vertex coloring problem have originally been studied in the context of frequency assignment, providing perfect showcases of the interaction between discrete mathematics and applications.

The combination of T- and list coloring is known as *List-T-coloring*. If not the number of frequencies but the difference between highest and lowest frequency has to be minimized, the problem is known as the *minimum span* FAP.

In the case of a fixed spectrum, K-COLORABILITY is generalized in two ways in case the problem is infeasible. If the planner needs to assign a frequency to each transmitter, some unacceptable interference has to be permitted. Otherwise, the planner might consider leaving some transmitters unassigned to avoid unacceptable levels of interference. If only co-channel interference is considered, the first case is known as the *minimum K-partition* problem: We associate a value $c_e^{co} \in \mathbb{Q}_+$ with all edges $e \in E$. The minimum K-partition problem now consists of a partition of the vertices in K subsets $S_1, \ldots, S_K$ such that the sum over all subsets of the total weight c_e^{co} of the edges in $G[S_i]$ is minimized. Stated otherwise, all vertices in a subset are assigned the same frequency, and so the interference incurred by this assignment is given by the sum of the weights c_{ij}^{co} for all $i, j \in S_i$. A solution with total weight 0 exists if and only if the graph is K-colorable. See [42] for a solution approach using *semidefinite programming*.

The second case is known as *k-colorable induced subgraph*. In this problem we have to color as many vertices as possible with K colors. A solution where all vertices are colored implies that the graph is K-colorable, whereas non-K-colorable graphs have at least one vertex uncolored in any K-colorable induced subgraph solution.

The inclusion of potential interference between nonidentical frequencies for transmitter pairs results in the study of the so-called *minimum blocking* FAP. Inclusion of the same information in the minimum K-partition problem results in the *minimum interference* FAP.

Frequency Assignment in GSM Networks

We conclude the discussion of FAP with the formulation as integer linear program of the minimum interference problem with co- and *adjacent-channel interference*, as it occurs frequently in GSM networks.[2] Usually an operator of a GSM cellular phone network has acquired the right to use a certain spectrum of frequencies $[f_{min}, f_{max}]$ in a particular geographical region, e.g., a country. The frequency band is—depending on the technology utilized—partitioned into a set of *channels*, all with the same bandwidth Δ. The available channels are here denoted by $F = \{1, 2, \ldots, N\}$, where $N = (f_{max} - f_{min})/\Delta$. A transmitter pair is exposed to adjacent-channel interference if the assigned frequencies are consecutive numbers in the spectrum.

In GSM networks, communication between mobile and base station (*up-link*) as well as between base station and mobile (*down-link*) must be established. To simplify the management of these networks, two separate, but paired, spectrums of frequencies are licensed, one for up-link and one for down-link. For FAP one can restrict assigning down-link frequencies to the base stations, as the paired frequencies can then be used for up-link communication.

We associate with the edges of the interference graph $G = (V, E)$ two values $c_{ij}^{co}, c_{ij}^{ad} \in \mathbb{Q}_+$ denoting the level of co- and adjacent-channel interference between

---

[2] Some details are ignored; see, for example, [43] for a more detailed discussion

the vertices. For ease of notation we introduce two subsets:

$$E^{co} = \{ij \in E \mid c^{co}_{(}ij) > 0\}, \text{ and}$$

$$E^{ad} = \{ij \in E \mid c^{ad}_{(}ij) > 0\}.$$

Given a subset $F_i \subseteq F$ for all $i \in V$, the formulation uses binary variables x_{if}, indicating whether frequency $f \in F_i$ is assigned to vertex $i \in V$. We further introduce variables z^{co}_{ij} and z^{ad}_{ij} to denote violation of the co-channel and adjacent-channel constraints, respectively. Then, the integer linear program reads

$$\min \quad \sum_{ij \in E^{co}} c^{co}_{ij} z^{co}_{ij} + \sum_{ij \in E^{ad}} c^{ad}_{ij} z^{ad}_{ij} \tag{1.27a}$$

$$\text{s.t.} \quad \sum_{f \in F_i} x_{if} = 1 \qquad \forall i \in V \tag{1.27b}$$

$$x_{if} + x_{jf} - z^{co}_{ij} \le 1 \qquad \forall ij \in E^{co}, f \in F_i \cap F_j \tag{1.27c}$$

$$x_{if} + x_{jg} - z^{ad}_{ij} \le 1 \qquad \forall ij \in E^{ad}, f \in F_i, g \in F_j: |f - g| = 1 \tag{1.27d}$$

$$x_{if}, z^{co}_{ij}, z^{ad}_{ij} \in \{0,1\} \tag{1.27e}$$

If $ij \in E^{co}$ and frequency f can be assigned to both vertices, constraint (1.27c) assures that such a mutual assignment incurs a penalty of c^{co}_{ij}. Similarly, constraint (1.27d) guarantees that the adjacent-channel interference is comprised of the objective if $ij \in E^{ad}$ and the frequencies f and g differ by 1.

A major drawback of the above formulation is the difficulty to solve these coloring-like integer linear programs to optimality. A wide range of different approaches have therefore been proposed to compute close-to-optimal solutions and/or lower bounds on the minimum total interference; see [3] for a survey. In Chapter 11 a similar frequency assignment model is studied in more detail for the case of wireless local area networks (WLANs). Due to the limited number of frequencies the drawbacks of the above formulation have less impact in this case.

Figure 1.14 shows an example of a carrier network, where the so-called DSATUR heuristic [24] followed by a very simple local search heuristic (1-opt) is applied. This procedure improved the solution provided by a network operator by 96% [23, 43]. In the plots, transmitters are represented by dots in the Euclidean plane according to their geographical coordinates. Two transmitters are connected by a colored line if the assignment of frequencies results in interference. If the line is drawn in a pale color, then the interference is small. With increasing interference, the color of the line turns black.

1.5.5.2 Maximization of Network Coverage

Besides the frequencies on which the antennae operate, the locations of the antennae and their configuration play an important role in the performance of the network. In fact, these decisions are typically taken before the frequencies are considered.

(a) operated frequency plan   (b) optimized frequency plan
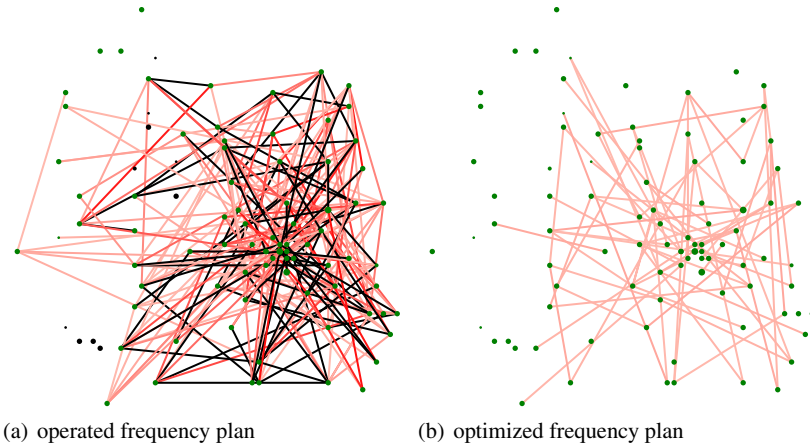
Fig. 1.14 Interference reduction of 96% by heuristics.

In the most simple version, the main objective is to maximize the area covered by the installed antennae, given a certain budget; or alternatively, to minimize the installation cost subject to the requirement of covering a certain percentage of the area, e.g., 99%.

To measure the coverage of a wireless network within a region (e.g., building, city center, county, country), a grid of pixels I representing small areas is laid out. A set of possible locations for antennae A is introduced, with cost parameter c_a for installing an (omnidirectional) antenna.

For each antenna location $a \in A$, we introduce a binary variable $x_a \in \{0, 1\}$ to denote whether or not an antenna is installed at this location. To model the constraints, we introduce sets $A_i \subseteq A$ defining which antennae locations would be able to cover pixel $i \in I$ (assuming a particular configuration of antenna $a \in A$). Now, the minimum cost 100% coverage problem reads

$$\min \quad \sum_{a \in A} c_a x_a \tag{1.28a}$$

$$\text{s.t.} \quad \sum_{a \in A_i} x_a \geq 1 \qquad \qquad \forall i \in I \tag{1.28b}$$

$$x_a \in \{0, 1\} \tag{1.28c}$$

This model precisely describes a minimum cost set covering problem for base set I with subsets $I_a = \{i \in I \mid a \in A_i\}$. Hence, the problem is NP-hard.

Coverage rates below 100% can also be modeled with an extra set of binary variables $y_i \in \{0, 1\}$ denoting whether or not a pixel is covered. If p denotes the percentage of pixels to be covered, the new model reads

$$\min \quad \sum_{a \in A} c_a x_a \tag{1.29a}$$

$$\text{s.t.} \quad \sum_{a \in A_i} x_a - y_i \geq 0 \qquad \forall i \in I \tag{1.29b}$$

$$\sum_{i \in I} y_i \geq p \, |I| \tag{1.29c}$$

$$x_a, y_i \in \{0,1\} \tag{1.29d}$$

Constraints (1.29b) now require an antenna installed close to pixel i in order to set y_i to one. Constraints (1.29c) ensure that $p\%$ of the pixels are covered.

An alternative is to maximize the network coverage, given a certain budget B for network installation cost:

$$\max \quad \sum_{i \in I} y_i \tag{1.30a}$$

$$\text{s.t.} \quad \sum_{a \in A_i} x_a - y_i \geq 0 \qquad \forall i \in I \tag{1.30b}$$

$$\sum_{a \in A} c_a x_a \leq B \tag{1.30c}$$

$$x_a, y_i \in \{0,1\} \tag{1.30d}$$

A final generalization to be discussed here is the inclusion of a bandwidth capacity constraint. For this, a traffic value d_i is estimated for each pixel $i \in I$ and an antenna capacity D is given. Since we now need to know which pixel is covered by which antenna, we have to introduce a new set of binary variables $z_{ia} \in \{0,1\}$ for all $i \in I$ and $a \in A_i$ to denote this assignment. Without further restrictions, the model 100% coverage problem can now be described as

$$\min \quad \sum_{a \in A} c_a x_a \tag{1.31a}$$

$$\text{s.t.} \quad \sum_{a \in A_i} z_{ia} = 1 \qquad i \in I \tag{1.31b}$$

$$\sum_{i \in I_a} d_i z_{ia} - D x_a \leq 0 \qquad a \in A \tag{1.31c}$$

$$x_a, z_{ia} \in \{0,1\} \tag{1.31d}$$

Constraints (1.31c) limit the number of test points assigned to an antennae on the basis of demand and capacity (and only if the antenna location is chosen). This problem is a special case of the (capacitated) *facility location problem* (the difference being that no assignment costs exist); see [62].

All above presented models are simplified in the sense that important issues in wireless networks are missing. A *cell* in a wireless network is defined as all pixels assigned to a single antenna. Cells are usually expected to have a certain degree of connectedness, but this aspect is completely ignored in the above models. In fact, the connectedness is (partly) implied by the signal strengths received at the pixel.

The strongest signal determines the allocation of the pixel to an antenna. This and other issues are studied in depth in Chapter 11. An approach combining frequency assignment and network coverage is also presented.

1.6 Emerging Applications in Communication Networks

The subsequent chapters of this book are devoted to the application of graph-theoretical concepts and algorithms to emerging communication networking applications. The chapters consider problems that have been studied in the context of the European COST action 293 – Graphs and Algorithms in Communication Networks [33]. According to the COST 293 working groups, Part I discusses topics in the area of Broadband and Optical Networks, including multilayer networks, whereas Part II describes studies in Wireless and Ad Hoc Networks. Here we provide a brief overview.

1.6.1 Broadband and Optical Networks

Chapter 2 presents a survey on theoretical and practical aspects of *traffic grooming* in *optical networks*. Traffic grooming was one of the key research topics within COST 293, and accordingly many participants have contributed to this chapter. The chapter was coordinated by David Coudert. In high-speed optical networks, every optical channel provides a huge amount of bandwidth. A single (aggregated) request usually does not require the whole bandwidth an optical channel provides. This implies an opportunity to save devices like light termination equipment (transmitters and receivers) and Add/Drop Multiplexers (ADMs) in order to reduce the investment and operation cost of the network. Chapter 2 surveys the main theoretical results for different grooming factors on various topologies, introduces an ILP formulation for the optimization problem on general topologies, and presents some experimental results.

Next, in Chapter 3 the closely related problem of multilayer network design is discussed. The chapter, coordinated by Sebastian Orlowski, introduces a two-layer network design problem which has been studied in collaboration with Nokia Siemens Networks. A mixed-integer programming formulation is presented that takes many practical side constraints into account, including node hardware, several bit rates, and survivability against single physical node or link failures. The model is solved by a branch-and-cut algorithm using problem-specific preprocessing, MIP-based heuristics, and cutting planes. The authors show that for realistic instances computation times can be significantly reduced compared to the black-box solution.

Multi-protocol Label Switching (MPLS) is one of the routing protocols in packet-oriented communication networks that uses labels attached to the packets to encode the path the packet should follow through the network. With the increasing num-

ber of network nodes to route packets to, it is important to keep the length of the labels small without losing their flexibility. Chapter 4 addresses this issue from an optimization point of view. Fernando Solano was the leading author of this chapter.

Chapter 5, coordinated by José L. Marzo and Thomas Stidsen, provides a showcase on the issue of survivability in networks. Instead of reserving dedicated paths to assure connections are not lost in the case of a link or node failure, one might consider the usage of a different set of protection paths for every failure situation. This chapter formalizes this concept, provides exact and heuristic algorithms for the online routing problem involved, and finally presents two case studies on the resource provisioning and connection availability for both dedicated and shared path protection in heterogeneous network topologies.

Besides connection-oriented and packet-oriented network technologies, optical burst switching (OBS) has been proposed as an intermediate solution for all-optical networks. In Chapter 6 Mirosław Klinkowski and his coauthors discuss routing optimization in such networks where buffering is in general not possible and hence the network is sensitive to congestion and traffic losses. Nonlinear optimization models for three network loss models are studied and solutions are presented that reduce the burst loss probability.

Next, the issue of dynamic bandwidth allocation is discussed in Chapter 7, led by Xavier Hesselbach. To maintain and ensure end-to-end in-sequence routing of packets, load balancing and bandwidth/flow allocation in MPLS-based architectures have to be established. Traffic characteristics such as Quality of Service (QoS) and burstiness are considered.

Chapter 8 discusses optimization problems in the context of the open shortest path (OSPF) routing protocol for Internet traffic. The main advantage of the shortest path routing policy is its simplicity, allowing for little administrative overhead. From the network engineering perspective, however, shortest path routing can pose problems in achieving satisfactory traffic handling efficiency. The chapter discusses one of the main tasks when planning such a network: the setting of the administrative weights of the links such that a globally efficient traffic routing is achieved. This very difficult optimization problem is considered, models are proposed, and exact and heuristic solution methods are discussed. The chapter was coordinated by Andreas Bley and Michał Pióro.

So far, it has been assumed in all chapters that a central authority is able to make decisions, or coordinate such decisions that determine the performance of the network. In Chapter 9 the scenario where such a central authority does not exist are considered. Network users act in an uncoordinated and selfish manner, affecting the overall system performance. Issues such as the definition of reasonable and practical models for studying this kind of behavior or the quantification of the efficiency loss due to the lack of users' cooperation are discussed, and results are presented. The chapter was coordinated by Ioannis Caragiannis.

Finally, Chapter 10 (by Ignasi Sau and Janez Žerovnik) studies the routing of packets in a timely fashion. The goal is to minimize the number of rounds a routing protocol needs to send a set of requests using a system of paths in the network. More information on the network topology (e.g., hexagonal grids) or the set of requests

(e.g., so-called permutation routing or *r*-central routing) allows for more accurate results. The chapter surveys the results in the general setting and in particular cases.

1.6.2 Wireless and Ad Hoc Networks

The part on wireless and ad hoc networks starts with a study on Wireless Local Area Network (WLAN) planning. Chapter 11, coordinated by Di Yuan, considers the optimization of such networks, taking into account performance measures like deployment cost, coverage, capacity, interference, data throughput, and radio resource utilization. Both individual models and an integrated model for placing access points and channel assignment are presented. Computational results are reported on the basis of real-life data.

Broadcasting is a basic network communication task, where a message initially held by a source node has to be disseminated to all other nodes in the network. Broadcasting in radio networks is studied in Chapter 12. David Peleg and Tomasz Radzik review the literature on time-efficient algorithms under a variety of models and assumptions. They also illustrate the basic techniques to prove the obtained results.

Chapter 13 studies similar problems from the energy consumption perspective. In wireless networks energy is a scarce resource since many network nodes do not have a fixed infrastructure (e.g., notebooks, PDAs, etc.). In this chapter, two problems are studied. The aim of *minimum energy broadcast routing* is to route a message from a given source to all other nodes with minimum overall energy usage. In the *Multi-interface Networks*, the energy consumption is minimized by the choice of activated interfaces on devices with multiple interfaces (e.g., Bluetooth, WiFi, etc.). The leading author of the chapter was Alfredo Navarra.

In contrast to broadcasting, in *data gathering* information provided or collected by the network nodes has to be gathered in a specific node. Due to interference, simultaneous data communication is limited and has to be carried out in a number of rounds. Hence, minimization of the number of rounds or similar objectives have to be accounted for. In Chapter 14, Ralf Klasing and his coauthors consider different interference models and discuss recent complexity results as well as approximation algorithms.

Chapter 15, by Jérôme Galtier, focuses on another aspect of WLANs: protocols for Medium Access Control (MAC). The transmission of a packet might fail due to the collision with simultaneously sent packets (from other sources). To achieve a good network performance, protocols are needed to handle the retransmission of the packet. In this chapter, mathematical evidence is presented that the *selective tournaments* protocol is asymptotically tight. Moreover, it is shown that in the context of WiFi and WiMAX networks a collision reduction of 14% to 21% can be achieved compared to the best known methods.

Finally, Chapter 16, coordinated by Lenka Carr-Motyckova and Walter Unger), discusses topology control and routing in *ad hoc networks*. Mobile entities with

the ability to communicate via radio signals may form an ad hoc network. Special problems arising for these ad hoc networks are considered briefly: range control, the reduction of interferences, regulation of power consumption, and localization.

References

1. Aardal, K. I., Hoesel, C. P. M. v.: Polyhedral techniques in combinatorial optimization I: theory. Statistica Neerlandica **50**(1), 3–26 (1996)
2. Aardal, K. I., Hoesel, C. P. M. v.: Polyhedral techniques in combinatorial optimization II: applications and computations. Statistica Neerlandica **53**(2), 131–177 (1999)
3. Aardal, K. I., Hoesel, C. P. M. v., Koster, A. M. C. A., Mannino, C., Sassano, A.: Models and solution techniques for the frequency assignment problem. Annals of Operations Research **153**, 79–129 (2007). URL http://dx.doi.org/10.1007/s10479-007-0178-0
4. Aarts, E. H. L., Lenstra, J. K. (eds.): Local Search in Combinatorial Optimization. Wiley, Chichester (1997)
5. Achterberg, T.: SCIP: solving constraint integer programs. Mathematical Programming Computation **1**(1), 1–41 (2009). URL http://scip.zib.de/
6. Agrawal, A., Klein, P., Ravi, R.: When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks. SIAM Journal on Computing **24**, 440–456 (1995)
7. Ahuja, R. K., Magnanti, T. L., Orlin, J. B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, New Jersey (1993)
8. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and approximation: combinatorial optimization problems and their approximability properties. Springer-Verlag (1999)
9. Bang-Jensen, J., Gutin, G.: Digraphs: Theory, Algorithms and Applications, 2 edn. Springer-Verlag (2008). http://www.cs.rhul.ac.uk/books/dbook/
10. Bansal, N., Khandekar, R., Nagarajan, V.: Additive guarantees for degree bounded directed network design. In: Proceedings of the 40th annual ACM symposium on Theory of computing, STOC 2008, pp. 769–778 (2008)
11. Barnhart, C., Hane, C. A., Vance, P. H.: Using branch-and-price-and-cut to origin-destination integer multi-commodity flow problems. Operations Research **48**, 318–326 (2000)
12. Bazaraa, M. S., Shetty, C. M.: Nonlinear programming: Theory and algorithms. John Wiley & Sons (1979)
13. Belady, L. A.: A study of replacement algorithms for a virtual-storage computer. IBM Systems Journal **5**(2), 78–101 (1966)
14. Bellare, M., Goldreich, O., Sudan, M.: Free bits, pcps and non-approximability – towards tight results. SIAM Journal on Computing **27**, 804–915 (1998)
15. Bermond, J. C., Delorme, C., Quisquater, J. J.: Strategies for interconnection networks: some methods from graph theory. Graphs and Combinatorics **5**, 107–123 (1989)
16. Bertsekas, D. P.: Nonlinear Programming, 2nd edn. Athena Scientific (1999)
17. Bhandari, R.: Survivable Networks: Algorithms for Diverse Routing. Kluwer Academic Publishers (1999)
18. Bienstock, D., Bley, A.: Capacitated network design with mulicast commodities. In: Proceedings of 8th International Conference on Telecommunication Systems. Nashville, TN (2000). URL http://www.zib.de/Publications/abstracts/ZR-00-14/
19. Bienstock, D., Chopra, S., Günlük, O., Tsai, C. Y.: Minimum cost capacity installation for multicommodity network flows. Mathematical Programming **81**, 177–199 (1998)

20. Bixby, R.: Solving real-world linear programs: A decade and more of progress. Operations Research **50**(1), 3–15 (2002)
21. Bixby, R. E., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: Mixed-Integer Programming: A Progress Report. In: M. Grötschel (ed.) The Sharpest Cut: The Impact of Manfred Padberg and His Work, *MPS-SIAM Series on Optimization*, vol. 4, pp. 309–326. SIAM (2004)
22. Bondy, J. A., Murty, U. S. R.: Graph Theory with Applications. North-Holland (1976). http://www.ecp6.jussieu.fr/pageperso/bondy/books/gtwa/gtwa.html
23. Borndörfer, R., Eisenblätter, A., Grötschel, M., Martin, A.: Frequency assignment in cellular phone networks. Annals of Operations Research **76**, 73–94 (1998)
24. Brélaz, D.: New methods to color the vertices of a graph. Communications of the ACM **22**, 251–256 (1979)
25. Chartrand, G., Lesniak, L.: Graphs & Digraphs. Wadsworth & Brooks (1986)
26. Chopra, S., Gilboa, I., Sastry, S.: Source sink flows with capacity installation in batches. DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science **85** (1998)
27. Chopra, S., Rao, M. R.: The Steiner tree problem I: Formulations, compositions and extension of facets. Mathematical Programming **64**, 209–229 (1994)
28. Chung, F. R. K., Coffman, E. G., Reiman, M. I., Simon, B.: On the capacity and forwarding index of communication networks. IEEE Trans. on Information Theory **33**, 244–232 (1987)
29. Chvátal, V.: Linear Programming. W. H. Freeman and Company, New York (1983)
30. Clos, C.: A study of non-blocking switching networks. Bell System Technical Journal (1953)
31. Cook, W.: Fifty-plus years of combinatorial integer programming (2009). URL http://www2.isye.gatech.edu/~wcook/ip50/ip50.pdf
32. Cook, W. J., Cunningham, W. H., Pulleybank, W. R., Schrijver, A.: Combinatorial Optimization. Series in Discrete Mathematics and Optimization. Wiley-Interscience (1998)
33. Cost action 293 – graphs and algorithms in communication networks. http://www.cost293.org/
34. Coudert, D., Ferreira, A., Muñoz, X.: A multihop-multi-OPS optical interconnection network. IEEE/OSA Journal of Lightwave Technology **18**(12), 2076–2085 (2000)
35. Coudert, D., Muñoz, X.: Graph theory and traffic grooming in WDM rings. In: Recent Research Developments in Optics, 3, chap. 37, pp. 759–778. Research Signpost. Kerala, India (2003)
36. Dantzig, G. B.: Linear Programming and Extensions. Princeton University Press (1998). Originally published in 1963
37. Dantzig, G. B., Fulkerson, D. R.: On the max-flow min-cut theorem of networks. In: H. W. Kuhn, A. W. Tucker (eds.) Linear Inequalities and Related Systems, *Annals of Mathematics Series*, vol. 38, pp. 215–221. Princeton University Press, Princeton, New Jersey (1956)
38. Diestel, R.: Graph Theory, *Graduate Texts in Mathematics*, vol. 173, 3 edn. Springer-Verlag, Heidelberg (2005). http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/
39. Dijkstra, E. W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**, 269–271 (1959)
40. Dolev, D., Halpern, J., Simons, B., Strong, H.: A new look at fault tolerant network routing. Information and Computation **72**, 180–196 (1987)
41. Eisenblätter, A.: Frequency assignment in GSM networks: Models, heuristics, and lower bounds. Ph.D. thesis, Technische Universität Berlin, Berlin, Germany (2001)
42. Eisenblätter, A.: The semidefinite relaxation of the k-partition polytope is strong. In: W. J. Cook, A. S. Schulz (eds.) Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO'02), *Lecture Notes in Computer Science*, vol. 2337, pp. 273–290. Springer-Verlag, Berlin Heidelberg (2002)
43. Eisenblätter, A., Grötschel, M., Koster, A. M. C. A.: Frequency planning and ramifications of coloring. Discussiones Mathematicae Graph Theory **22**(1), 51–88 (2002)
44. Eisenblätter, A., Koster, A. M. C. A.: FAP web – A website devoted to frequency assignment. URL: http://fap.zib.de (2000–2008)

45. Fàbrega, J., Fiol, M. A.: Maximally connected digraphs. Journal of Graph Theory **13**, 657–668 (1989)

46. Ferreira, A., Pérennes, S., Richa, A. W., Rivano, H., Moses, N. S.: Models, Complexity and Algorithms for the Design of Multifiber WDM Networks. Telecommunication Systems **24**, 123–138 (2003)

47. Fiat, A., Karp, R. M., Luby, M., McGeoch, L. A., Sleator, D. D., Young, N. E.: Competitive paging algorithms. Journal of Algorithms **12**(4), 685–699 (1991)

48. Fortz, B., Labbé, M.: Design of survivable networks. In: M. G. C. Resende, P. M. Pardalos (eds.) Handbook of Optimization in Telecommunications, chap. 15, pp. 367–389. Springer (2006)

49. Garey, M. R., Johnson, D. S.: The rectilinear Steiner tree problem is NP-complete. SIAM Journal on Applied Mathematics **32**, 826–834 (1977)

50. Garey, M. R., Johnson, D. S.: Computers and intractability: a guide to the theory of NP-completeness. Freeman and Company, N. Y. (1979)

51. Goemans, M. X.: Minimum bounded degree spanning trees. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006, pp. 273–282 (2006)

52. Grötschel, M., Krumke, S. O., Rambau, J. (eds.): Online Optimization of Large Scale Systems. Springer (2001)

53. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**, 169–197 (1981)

54. Grötschel, M., Lovász, L., Schrijver, A.: Corrigendum to our paper "the ellipsoid method and its consequences in combinatorial optimization". Combinatorica **4**, 291–295 (1984)

55. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. No. 2 in Algorithms and Combinatorics. Springer-Verlag (1988)

56. Grötschel, M., Monma, C. L., Stoer, M.: Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. SIAM Journal on Optimization **2**(3), 474–504 (1992)

57. Grötschel, M., Monma, C. L., Stoer, M.: Design of Survivable Networks. In: M. O. Ball, T. L. Magnanti, C. L. Monma, G. L. Nemhauser (eds.) Network Models, *Handbooks in Operations Research and Management Science*, vol. 7, pp. 617–672. North-Holland (1995)

58. Grötschel, M., Monma, C. L., Stoer, M.: Polyhedral and Computational Investigations for Designing Communication Networks with High Survivability Requirements. Operations Research **43**(6), 1012–1024 (1995)

59. Grover, W. D.: Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking. Prentice Hall (2003)

60. Henningsson, M., Holmberg, K., Yuan, D.: Ring network design. In: M. G. C. Resende, P. M. Pardalos (eds.) Handbook of Optimization in Telecommunications, chap. 12, pp. 291–311. Springer (2006)

61. Heydemann, M. C., Meyer, J. C., Sotteau, D.: On forwarding indices of networks. Discrete Applied Mathematics **23**, 103–123 (1989)

62. Holmberg, K., Rönnqvist, M., Yuan, D.: An exact algorithm for the capacitated facility location problems with single sourcing. European Journal of Operational Research **113**(3), 544–559 (1999)

63. Hromkovic, J., Klasing, R., Monien, B., Peine, R.: Dissemination of information in interconnection networks (broadcasting and gossiping) (1996)

64. ILOG: CPLEX version 11.1 (2008). http://www.ilog.com/products/cplex

65. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica **4**(4), 373–395 (1984)

66. Karp, R. M.: Reducibility among combinatorial problems. In: R. E. Miller, J. W. Thatcher (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)

67. Klee, V., Minty, G. J.: How good is the Simplex algorithm? In: O. Shisha (ed.) Inequalities, III, pp. 159–175. Academic Press, New York, NY (1972)

68. Koch, T., Martin, A., Voß, S.: SteinLib: An updated library on Steiner tree problems in graphs. Tech. Rep. ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin (2000). URL `http://elib.zib.de/steinlib`
69. Kruskal Jr., J. B.: On the shortest spanning tree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society **7**, 48–50 (1956)
70. LEMON: A C++ Library for Efficient Modelling and Optimization in Networks. `http://lemon.cs.elte.hu`
71. Lindberg, P.: Network optimisation with successive smooth approximation. In: 11th Nordic Teletraffic Seminar. Stockholm (1993)
72. Lynch, N.: Distributed Algorithms. Morgan Kaufmann Publishers (1996)
73. Marsden, G., Marchand, P., Harvey, P., Esener, S.: Optical transpose interconnection system architectures. OSA Optics Letters **18**(13), 1083–1085 (1993)
74. Martello, S., Toth, P.: Knapsack Problems – Algorithms and Computer Implementations. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons (1990). URL `http://www.or.deis.unibo.it/kp.html`
75. Metzger, B. H.: Spectrum management technique (Fall 1970). Presentation at 38th National ORSA meeting (Detroit, MI)
76. Nemhauser, G. L., Wolsey, L. A.: Integer and Combinatorial Optimization. Wiley, N. Y. (1988)
77. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0–Survivable Network Design Library. In: Proceedings of International Network Optimization Conference (INOC2007) (2007). `http://sndlib.zib.de`
78. Peleg, D.: Distributed Computing – A Locality-Sensitive Approach. Monographs in Discrete Mathematics and Applications. SIAM (2000)
79. Pióro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Morgan Kaufmann (2004)
80. Polzin, T.: Algorithms for the Steiner problem in networks. Ph.D. thesis, Universität des Saarlandes (2003)
81. Prim, R. C.: Shortest connection networks and some generalizations. The Bell System Technical Journal **36**, 1389–1401 (1957)
82. Schrijver, A.: Theory of linear and integer programming. Wiley, New York (1986)
83. Schrijver, A.: Combinatorial Optimization – Polyhedra and Efficiency. No. 24 in Algorithms and Combinatorics. Springer (2003)
84. Sleator, D. D., Tarjan, R. E.: Amortized efficiency of list update and paging rules. Communications of the ACM **28**(2), 202–208 (1985)
85. Stidsen, T., Petersen, B., Rasmussen, K. B., Spoorendonk, S., Zachariasen, M., Rambach, F., Kiese, M.: Optimal routing with single backup path protection. In: Proceedings International Network Optimization Conference (INOC 2007). Spa, Belgium (2007)
86. Suurballe, J. W.: Disjoint paths in a network. Networks **4**(2), 125–145 (1974)
87. Tel, G.: Introduction to Distributed Algorithms. Cambridge University Press (1994)
88. Vanderbei, R. J.: Linear Programming: Foundations and Extensions, *International Series in Operations Research & Management Science*, vol. 114, 3 edn. Springer-Verlag (2008)
89. Voš, S.: Steiner tree problems in telecommunications. In: M. G. C. Resende, P. M. Pardalos (eds.) Handbook of Optimization in Telecommunications, chap. 18, pp. 459–515. Springer (2006)
90. Wessäly, R.: DImensioning Survivable Capacitated NETworks. Ph.D. thesis, Technische Universität Berlin (2000)
91. West, D. B.: Introduction to Graph Theory, 2 edn. Prentice Hall (2001)
92. Wolsey, L. A.: Integer Programming. Series in Discrete Mathematics and Optimization. Wiley-Interscience (1998)
93. Zymolka, A.: Design of survivable optical networks by mathematical optimization. Ph.d. thesis, TU Berlin (2007)
94. Zymolka, A., Koster, A. M. C. A., Wessäly, R.: Transparent optical network design with sparse wavelength conversion. In: Proceedings of ONDM 2003, pp. 61–80. The 7th IFIP Working Conference on Optical Network Design & Modelling, Budapest, Hungary (2003)

Chapter 2
Traffic Grooming: Combinatorial Results and Practical Resolutions

Tibor Cinkler, David Coudert, Michele Flammini, Gianpiero Monaco, Luca Moscardelli, Xavier Muñoz, Ignasi Sau, Mordechai Shalom, and Shmuel Zaks

Abstract In an optical network using the wavelength division multiplexing (WDM) technology, routing a request consists of assigning it a route in the physical network and a wavelength. If each request uses $1/g$ of the bandwidth of the wavelength, we will say that the grooming factor is g. That means that on a given edge of the network we can groom (group) at most g requests on the same wavelength. With this constraint the objective can be either to minimize the number of wavelengths (related to the transmission cost) or minimize the number of Add/Drop Multiplexers (shortly ADM) used in the network (related to the cost of the nodes).

Here, we first survey the main theoretical results obtained for different grooming

Tibor Cinkler

Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary, e-mail: cinkler@tmit.bme.hu

David Coudert

MASCOTTE, INRIA, I3S, CNRS UMR6070, University of Nice-Sophia Antipolis, France, e-mail: David.Coudert@sophia.inria.fr

Ignasi Sau

MASCOTTE, INRIA, I3S, CNRS UMR6070, University of Nice-Sophia Antipolis, France, and Graph Theory and Combinatorics Group, Department of Applied Mathematics IV, Universitat Politècnica de Catalunya, Barcelona, Spain, e-mail: Ignasi.Sau@sophia.inria.fr

Michele Flammini, Gianpiero Monaco, Luca Moscardelli

Dipartimento di Informatica, Università degli Studi dell'Aquila, Via Vetoio, Loc. Coppito, 67100 L'Aquila, Italy, e-mail: flammini,gianpiero.monaco,moscardelli@di.univaq.it

Xavier Muñoz

Graph Theory and Combinatorics Group, Department of Applied Mathematics IV, Universitat Politècnica de Catalunya, Barcelona, Spain, e-mail: xml@ma4.upc.edu

Mordechai Shalom

Tel-Hai Academic College, Upper Galilee, 12210, Israel, e-mail: cmshalom@cs.technion.ac.il

Shmuel Zaks

Computer Science Department, Technion, Haifa, Israel, e-mail: zaks@cs.technion.ac.il

factors on various topologies: complexity, (in)approximability, optimal construc-
tions, approximation algorithms, heuristics, etc. Then, we give an ILP formulation
for multilayer traffic grooming and present some experimental results.

Key words: WDM networks, grooming, ADM, complexity, approximation algo-
rithms, heuristics, integer linear programming

2.1 Introduction

Traffic grooming refers to techniques used to organize and simplify routing and
switching in connection-oriented networks, such as WDM (*wavelength division
multiplexing*) or MPLS (*Multi-protocol Label Switching*) networks, in order to im-
prove the usage of the bandwidth and of the components, and therefore to reduce
the network cost.

   Typically, when establishing a connection in an optical network, one has to install
some equipment at both extremities of the connection, that is, an optical transmitter
(laser) at its source and an optical receiver at its destination. But due to the cost of
building, installing, and maintaining devices, it is usually more interesting to use a
single kind of device that can handle both transmission and reception. Such devices
are called *Light Termination Equipment*, or LTE for short. Thus, every connection
will involve two distinct LTEs, and two distinct connections may share the same
LTE, provided that one ends at a node while the other starts from that same node.
In this context, traffic grooming refers to minimizing the number of LTEs that are
needed in the network to serve all connection requests. The problem of minimizing
the number of LTEs in the network being NP-Hard [58, 84], research effort has con-
centrated on the development of efficient approximation algorithms for both static
and online traffic [52, 59, 64, 65, 103]. This is the subject of Section 2.3.

   At another level in the network, traffic grooming also refers to techniques used
to combine low-speed traffic streams onto high speed wavelengths in order to mini-
mize the network-wide cost in terms of electronic switching. Typically, nodes of the
network insert and/or extract the data streams on a wavelength by means of add/drop
multiplexers (ADMs). A WDM or DWDM (*dense WDM*) optical network can han-
dle many wavelengths, each with large bandwidth available. On the other hand, a
single user seldom needs such large bandwidth. Therefore, by using multiplexed
access such as TDMA (time-division multiple access) or CDMA (code-division
multiple access), different users can share the same wavelength, thereby optimiz-
ing the bandwidth usage of the network. By using traffic grooming, not only is the
bandwidth usage optimized, but also the cost of the network can be cut by reduc-
ing the total number of ADMs. Such techniques become increasingly important for
emerging network technologies, including SONET/WDM rings and MPLS/MPλS
backbones [108], for which traffic grooming is essential.

   In this context, one ADM is needed in a node each time we want to add or drop
traffic to or from a wavelength. Therefore, one has to place one ADM in a node for

each wavelength in which traffic is added or dropped, as can be seen in Figure 2.1. Here, the bandwidth requirement of a traffic stream is expressed as a fraction of the bandwidth offered by a single wavelength, which we call the *grooming factor*, *g*, and an ADM is able to drop (or add) up to *g* unitary traffic streams from (or to) a given wavelength. Thus, the traffic grooming problem is to minimize the total number of ADMs to be installed in the network in order to accomodate all traffic streams.

Given the general traffic grooming problem of minimizing the total number of ADMs to be installed in the network with respect to the traffic requirement being NP-complete [21, 101], recent works focus on specific issues. Most of the algorithms aim at grooming traffic in such a way that all the traffic between any given pair of nodes is carried on a minimum number of wavelengths. However, a large part of the network cost depends on the capacity of the multiplexing equipment required at each node. Hence, in order to minimize the overall network cost, algorithms have to take into account a trade-off between the number of wavelengths used and the number of required ADMs. Indeed minimizing the number of ADMs is different from minimizing the number of wavelengths: the number of wavelengths and the number of ADMs cannot always be simultaneously minimized (see [11, 21, 69] for unitary traffic). Both minimization problems have been considered by many authors. See, for example, the surveys [3, 56] for minimization of the number of wavelengths and [10, 69, 70, 73, 112, 115] for minimization of ADMs, and [72, 81] for online approaches. Numerical results, heuristics, and tables might be found in [11, 113], and extensions to multicast connection requests in [51, 107].

The reader may also refer to the surveys [27, 57, 89, 117] and books [55, 106, 118] for other aspects of traffic grooming that are not considered here; in particular, *waveband switching* allows switching together a set of predetermined wavelengths (band) issued from one fiber and going to another [18–20, 75, 116]. Various other concepts might also been considered as traffic grooming, such as Lighttrails [114], Lighttours [105], or bus labeling [16, 17].

In this chapter, we give an overview of the traffic grooming problems that have been addressed within the European project COST 293 GRAAL, and we survey the main exact and approximate results obtained so far for static and online traffic. We present practical approaches for multilayer traffic grooming. The results have been obtained using a large variety of mathematical tools including graph theory, design theory, linear programming, combinatorial optimization, and game theory.

This chapter is structured as follows. We start in Section 2.2 with a general definition of the traffic grooming problem, and we give some examples. In Section 2.3 we present the modelization and the main results obtained for minimizing the number of LTEs in a network. We continue in Section 2.4 with the more general model of minimizing the number of ADMs, for which we survey the main combinatorial results. Then, in Section 2.5, we present an efficient ILP model for multilayer traffic grooming on general networks subject to general traffic demands. We also present some experimental results. We finally conclude this chapter in Section 2.6.
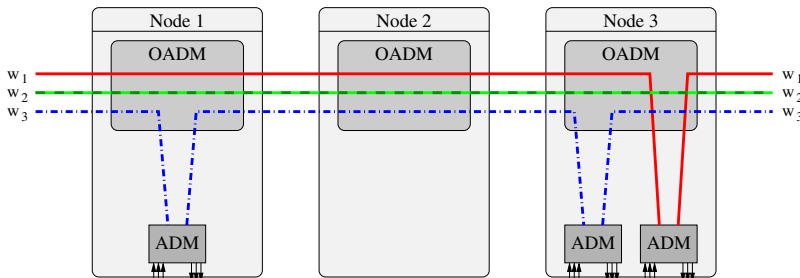
Fig. 2.1 Placement of ADMs in the network: one ADM for each wavelength used in a node

2.2 Problem Definition and Examples

In this section, we first give precise descriptions and models of LTE and ADM, and then formalize the traffic grooming problem considered here.

A Light Termination Equipment, LTE, is a device that realizes the interface between the optical domain and the electronic domain. It is constituted of one optical receiver and one optical transmitter, so every connection involves two distinct LTEs, one at each endpoint. In this chapter, we assume that the receiver and the transmitter of an LTE are tuned on the same wavelength (other assumptions are possible). Also, two distinct connections may share an LTE, provided that one ends at a node while the other starts from that same node, and that both connections are assigned the same wavelength.

An Add/Drop Multiplexer, ADM, is a device used in synchronous transmission networks (SDHs or SONETs) to add (insert) or drop (remove) lower-data-rate channel traffic from the higher-rate aggregated channel. In optical networks, each ADM contains an LTE to realize the interface between the optical domain (high-speed channel) and the electronic domain (lower-speed channels). Thus, an ADM operates on a single high-speed data stream, and so a single wavelength, as can be seen in Figure 2.1. The cost of an ADM is given by its capacity, that is, the maximum number of low-speed channels (provided each of them has a unitary bandwidth requirement) that can be added or dropped from the wavelength. The capacity of an ADM is called the *grooming factor* or *grooming ratio*. Finally, note that with grooming factor 1, an ADM is nothing other than an LTE.

In optical networks with grooming capabilities, the traffic demands are expressed in terms of low-speed data channels. Thus, one has to assign to each connection request a path and a wavelength with the capacity constraint that at most g (grooming factor) connection requests are assigned the same wavelength on the same link of the network.

An instance of the *traffic grooming problem* is a triple (G, I, g) where $G = (V, E)$ is a graph modeling the network topology, I is a set of connection requests, and g is a positive integer, namely the grooming factor.

Given a connection request $r \in I$ identified by a couple of nodes aiming to communicate, let P_r be the set of the paths in G connecting the two endpoints relative to r. We have two main issues:

- the determination of a path system (or path assignment) of (G,I), that is, a function $p : I \mapsto \bigcup_{r \in I} P_r$;
- the determination of a proper coloring (or wavelength assignment) of (G,I), that is, a function $w : I \mapsto N^+ = \{1,2,...\}$ such that for any edge $e \in E$ at most g paths using e are colored with the same color.

Some of the results presented in this chapter deal with both issues (Section 2.5), while others, given a path system in the input, focus only on the determination of a proper coloring (Sections 2.3 and 2.4).

Every colored request $r \in I$ needs an ADM at each of its endpoint nodes. Following the above description of ADMs, given a grooming factor g, at most g paths with the same color, incident to a node through the same edge, can use the same ADM. Furthermore, the same ADM can also be shared by at most g paths with the same color, incident to the same node through another incident edge.

The traffic grooming problem is the optimization problem of finding a proper coloring w of (G,I,g) minimizing the total number of ADMs used. Let $A(G,I,g)$ be the optimal value for such a problem.

To establish ideas we now provide two examples, for uni- and bidirectional rings, respectively.

Unidirectional Ring

Suppose we have a unidirectional ring with four nodes $\{1,2,3,4\}$ and an all-to-all unitary traffic (one request between each pair of nodes). Since we need one ADM at each extremity of a request, and the routing is unique, we can put requests (i,j) and (j,i) on the same wavelength, thus using $1/g$ of the capacity of that wavelength on the ring. We call such pair of symmetric requests a circle. There are therefore six circles (i,j) for $1 \leq i < j \leq 4$. If there is no grooming (i.e., $g=1$) we need six wavelengths (one per circle) and a total of 12 ADMs. If we have a grooming factor $g=2$, we can put on the same wavelength two circles, using three or four ADMs according to whether they share an end node or not. For example, we can put together $(1,2)$ and $(2,3)$ on one wavelength; $(1,3)$ and $(3,4)$ on a second wavelength; and $(1,4)$ and $(2,4)$ on a third wavelength, for a total of nine ADMs, and this is optimal.

Now, if we allow a grooming factor $g=3$, we can use only two wavelengths. If we put together on one wavelength $(1,2)$, $(2,3)$, and $(3,4)$ and on the other $(1,3)$, $(2,4)$, and $(1,4)$, we need eight ADMs (solution a, Figure 2(a)); but we can do better by putting on the first wavelength $(1,2)$, $(2,3)$ and $(1,3)$ and on the second $(1,4)$, $(2,4)$ and $(3,4)$, using seven ADMs (solution b, Figure 2(b)).

More formally, in the above example with $N=4$ and $g=3$, solution a consists of a decomposition of K_4 (all circles) into two paths with four vertices each,

(a) Solution with eight ADMs. Circles (1,2), (2,3), and (3,4) on the first
wavelength, and (1,3), (2,4), and (1,4) on the second wavelength

(b) Solution with seven ADMs. Circles (1,2), (2,3), and (1,3) on the first
wavelength, and (1,4), (2,4), and (3,4) on the second wavelength

(c) Decomposition of K_4 associated with the two solutions. Each edge of K_4 corre-
sponds to a circle
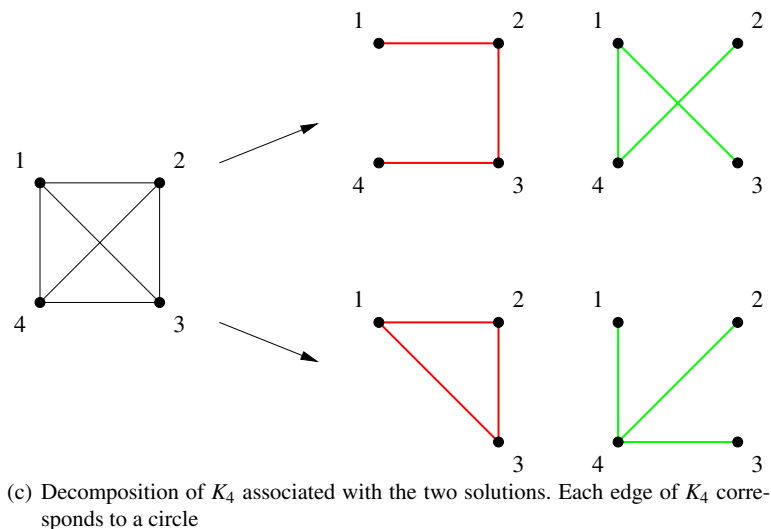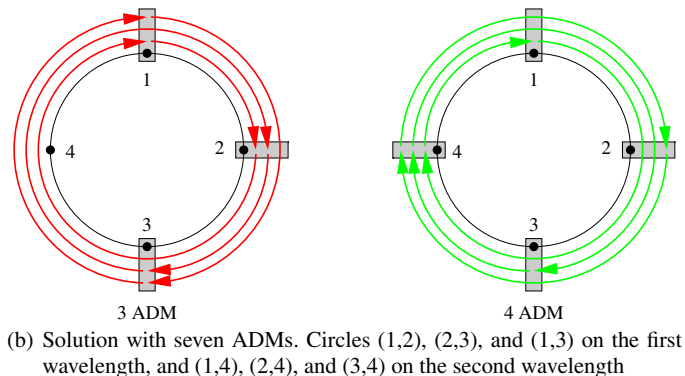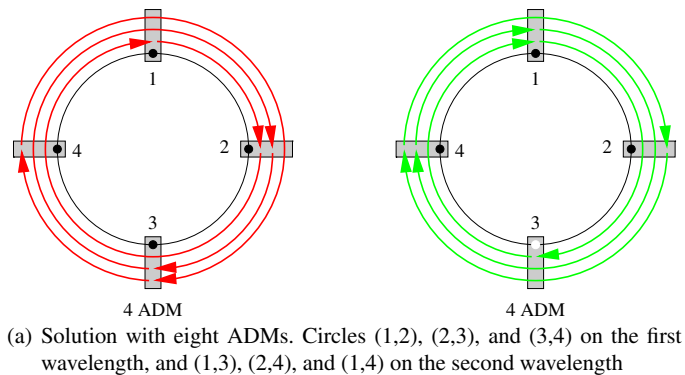
Fig. 2.2 Traffic grooming for a unidirectional ring with four nodes, grooming factor $g = 3$ all-to-all
unitary traffic. Solution 2(a) with eight ADMs, solution 2(b) with seven ADMs, and corresponding
decompositions of K_4

$[1,2,3,4]$ and $[1,4,2,3]$, while solution b corresponds to a decomposition into a triangle $(1,2,3)$ and a star with edges $(1,4)$, $(2,4)$, and $(3,4)$.

Bidirectional Ring

Consider now a bidirectional ring on five nodes $\{0,1,2,3,4\}$ with all-to-all unitary traffic modeled by the complete symmetric digraph K_5^+. In this setting, it is more interesting to route requests (i,j) and (j,i) on different wavelengths with shortest path routing. For example, with grooming factor $g = 3$, we can put on a wavelength routed clockwise requests $(i,i+1 \bmod 5)$ and $(i,i+2 \bmod 5)$, and on a wavelength routed counterclockwise requests $(i,i-1 \bmod 5)$ and $(i,i-2 \bmod 5)$. We need five ADMs on each wavelength so overall ten ADMs. But if requests (i,j) and (j,i) are routed on a same wavelength, then we can put at most three circles (pairs of symmetric requests) per wavelength using at least three ADMs. Since K_5^+ contains ten circles, we need four wavelengths, three of them with three circles and at least three ADMs and one of them with at least one circle and two ADMs, so overall 11 ADMs.

With grooming factor $g = 2$, we can put on one wavelength requests $(i,i+1 \bmod 5)$ and on another wavelength requests $(i,i+2 \bmod 5)$. Symmetric requests are routed similarly in opposite direction and we obtain the partition of Figure 3(b) using overall 20 ADMs. But we can do better by putting on a first wavelength requests $(i,i+1 \bmod 5)$, $(0,2)$ and $(2,4)$ using five ADMs, and on a second wavelength requests $(1,3)$, $(3,5)$, and $(4,1)$ using four ADMs. We obtain the partition of Figure 3(c) using overall $2(5+4) = 18$ ADMs.

(a) Set of requets

(b) Partition using two times ten ADMs

(c) Partition using two times nine ADMs
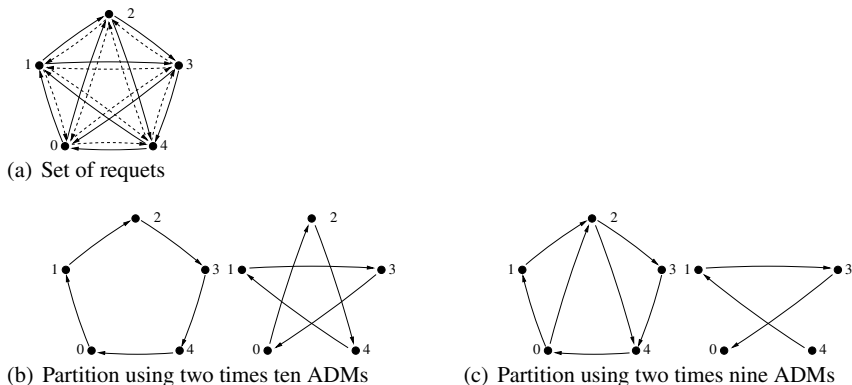
Fig. 2.3 Two valid partitions of K_5^+ when $g = 2$, using different number of ADMs. Symmetric requests are routed counterclockwise and partitioned similarly.

2.3 Minimizing the Usage of Light Termination Equipment

In this section, we concentrate on the traffic grooming problem of minimizing the total number of LTEs that are needed in the network to serve all connection requests. This problem is NP-hard [58, 84] in general but can be solved in polynomial time for specific topologies. Also, efficient approximation algorithms have been proposed for both static and online traffic.

This section is organized as follows. We first consider the path topology where the problem can be solved in polynomial time. Then we review efficient approximation algorithms for the ring topology where the problem is already NP-hard, and also for more general topologies. Finally, we show how game theory can be useful to solve dynamic and online versions of the problem.

2.3.1 Path

Let the physical topology be the directed path P_N with N nodes labelled $1, 2, \ldots, N$, and $N-1$ arcs $(i, i+1)$ for $1 \leq i < N$. Let also $TT_N = \{(i, j), \ 1 \leq i < j \leq N\}$ denote a *transitive tournament*, that is, the set of all requests from left to right.

For any set of requests $I \subseteq \lambda TT_N$, where λ is a positive integer, the problem of minimizing the number of LTEs on P_N can be solved optimally in polynomial time using a greedy algorithm. To prove that, it is sufficient to observe that the number of LTEs needed at node i of P_N is equal to $\max\{d_I^-(i), d_I^+(i)\}$, where $d_I^-(i)$ (or $d_I^+(i)$) denote the indegree (or outdegree) of node i in I, that is, the number of requests $\{u, i\}$ with $u < i$ (or $\{i, v\}$ with $i < v$). We obtain Proposition 2.1, and in Corollary 2.1 we give the exact number of LTEs when $I = TT_N$.

Proposition 2.1 (Bermond et al. [4]). $A(P_N, I, 1) = \sum_{i=0}^{N-1} \max\{d_I^-(i), d_I^+(i)\}$.

Corollary 2.1 (Bermond et al. [4]). $A(P_N, TT_N, 1) = \frac{3N^2 - 2N - \varepsilon}{4}$, *where* $\varepsilon = N \bmod 2$.

When the physical topology is a bidirectional path, it is necessary to be precise about how LTEs can be used. In particular, one has to be precise about whether it is possible to share a LTE between requests (u, i) and (i, v) with $u, v < i$, that is, a left-to-right request ending at i and a right-to-left request starting from i, or not. If it is not possible, then the problem can be decomposed into two subproblems on a directed path that will be solved independently. But when such sharing is allowed, the problem has not been addressed in the literature and it is conjectured to be NP-complete.

2.3.2 Ring

The problem of minimizing the number of LTEs in optical networks was introduced in [69] for the unidirectional ring topology. It is proved to be NP-hard independently in [58] and [84]. The NP-hardness proofs also apply to bidirectional rings, even when the routing of connection requests is given. An algorithm with approximation ratio of $\frac{3}{2}$ was presented in [52] for unidirectional and bidirectional rings with given routing. This algorithm has a first step (called the preprocessing step) that finds cycles in the instance and colors each cycle with a unique color. The remaining requests are then merged to form chains. This algorithm can also be adapted to the case where also the routing has to be determined, with the same approximation ratio [52].

This technique was improved in [103], showing that if the preprocessing phase tries to remove short cycles first, then an approximation ratio of $10/7 + \varepsilon$ can be achieved. This is improved to $10/7$ in [59] using the same technique with a more detailed analysis.

In [13], we give exact algorithms for the all-to-all set of requests on uni- and bidirectional rings. Surprisingly, these results are obtained using a partition of the set of requests into cycles of lengths 3 and 4.

In [53] and [60] a variant of this problem is considered. In this variant, a path can be broken into segments and each segment can be colored using a different wavelength. Obviously this might incur an additional cost in terms of LTEs, but it allows to reduce significantly the number of wavelength used.

2.3.3 General Topology

In [52] an approximation algorithm was presented for general networks. The algorithm has a preprocessing phase where cycles of length at most l are included in the solution; this algorithm was shown to have performance guarantee of $OPT + \frac{1}{2}(1 + \varepsilon)N, 0 < \varepsilon \le \frac{1}{l+2}$, where OPT is the cost of an optimal solution and N is the number of connection requests for any given odd l. A special case of this algorithm is when there is no preprocesing (i.e., $l = 1$). The analysis reduces in this case to $OPT + \frac{2}{3}N$. The dominant part in the running time of the algorithm is the preprocessing phase, which is exponential in l.

In [65] we improve the analysis of the algorithm of [52] and prove a performance of $OPT + \frac{1}{2}(1 + \varepsilon)N$, where $\frac{1}{2l+3} \le \varepsilon \le \frac{1}{\frac{3}{2}(l+2)}$. Specifically, we show that the algorithm guarantees an upper bound of $OPT + \frac{1}{2}(1 + \varepsilon)N$ for $\varepsilon \le \frac{1}{\frac{3}{2}(l+2)}$, and we demonstrate a family of instances for which the performance of the algorithm is $OPT + \frac{1}{2}(1 + \varepsilon)N$ for $\varepsilon \ge \frac{1}{2l+3}$.

Our analysis sheds more light on the structure and properties of the algorithm by closely examining the structural relation between the solution found by the algorithm and an optimal solution for any given instance of the problem. As the running

time of the algorithm is exponential in l, our result implies an improvement in the analysis of the running time of the algorithm. For any given $\varepsilon > 0$, the exponent of the running time needed to guarantee the approximation ratio $(3 + \varepsilon)/2$ is reduced by a factor of $3/2$. In addition, in the development of our bounds we address a purely combinatorial problem, which is of interest by itself.

We also improve the analysis for the special case where there is no preprocesing. In [64] we develop a new technique for the analysis of the upper bound and prove a tight bound of $OPT + \frac{3}{5}N$ for the performance of this algorithm.

2.3.4 Online Traffic

In many applications the requests arrive at the network online, and we have to assign them wavelengths so as to minimize the switching cost. In more involved cases we have also to determine the actual routing for these requests. In these situations, once an assignment is made the system cannot change it, and the aim is to suggest a strategy that will optimally utilize the network resources. Such a study is thus of great importance in the operation of optical networks.

Formally, an online algorithm is said to be c-competitive if, for any sequence of inputs, the cost is at most c times that of an optimal off-line algorithm (see [15]).

In [102] we present an online algorithm for the problem of minimizing the number of LTEs, and prove that its competitive ratio is $\frac{7}{4}$. We show that this result is the best possible in general. Moreover, we show that even for the ring topology network there is no online algorithm with competitive ratio better than $\frac{7}{4}$. We show that on the path topology the competitive ratio of the algorithm is $\frac{3}{2}$. This is the best possible for this topology. The lower bound on the ring topology does not hold when the ring is of bounded size. We analyze the triangle topology and show a tight bound of $\frac{5}{3}$ for it. The analysis of the upper bounds, as well as those for the lower bounds use all a variety of proof techniques, which are of interest on their own, and which might prove helpful in future research on the topic.

2.3.5 Price of Anarchy

Game Theory and the associated concept of Nash equilibria have recently emerged as a powerful tool for modeling and analyzing a lack of coordination in distributed environments. In this setting, each communication request is handled by an agent (or player) selfishly performing *moves*, i.e., changing her routing strategy in order to maximize her own benefit. A Nash equilibrium is a solution of the game in which no agent gains by unilaterally changing her routing strategy. If Nash equilibria are reached in a polynomial number of selfish moves, and finding an improving user move is a problem solvable in polynomial time, such a non-cooperative process naturally defines a distributed polynomial-time algorithm. However, due to the lack

of cooperation among the players, Nash equilibria are known not to always optimize the overall performance. Such a loss has been formalized by the so-called *price of anarchy* (or *optimistic price of anarchy*), defined as the ratio between the cost of the worst (or best) Nash equilibrium and the one of a centralized optimal solution. The notion of Nash equilibria goes back to [91]. For about a decade the use of game theory has gained a lot of attention in numerous computer science directions, in what is known today as algorithmic game theory (see [92, 100]). The notion of *price of anarchy* goes back to [82].

In [54] we consider non-cooperative games in all-optical networks where users share the cost of the LTE switches used for realizing given communication patterns. We show that the two fundamental cost sharing methods, Shapley and Egalitarian, induce polynomial converging games with price of anarchy at most $\frac{5}{3}$, regardless of the network topology. Such a bound is tight even for rings. Then, we show that if collusion of at most k players is allowed, the Egalitarian method yields polynomially converging games with price of collusion between $\frac{3}{2}$ and $\frac{3}{2} + \frac{1}{k}$. This result is very interesting and quite surprising, as the best-known approximation ratio, that is $\frac{3}{2} + \varepsilon$, can be achieved in polynomial time by uncoordinated evolutions of collusion games with coalitions of increasing size.

Moreover, with respect to the optimization of the optical spectrum, in [14] we investigate the problem in which a provider must determine suitable payment functions for non-cooperative agents wishing to communicate so as to induce routings in Nash equilibria using a low number of wavelengths. We assume three different information levels specifying the local knowledge that agents may exploit to compute their payments. Under complete information of all the agents and their routing requests, the network provider can compute prices where a Nash equilibrium is reached such that the assignment is the same as the one computed by a centralized algorithm. If the price to an agent is based only on the wavelengths used along connecting paths (minimal level) or along the edges (intermediate level), the most reasonable functions either do not admit equilibria or admit equilibria with the worst possible price of anarchy, that is, the ratio between the number of colors used by the worst Nash equilibrium and the one used by an optimal solution. However, by suitably restricting the network topology, a constant price of anarchy for chains and rings and a logarithmic one for trees have been obtained under the minimal and intermediate levels, respectively.

For more information, we refer to Chapter 9.

2.4 Minimizing the Number of Add/Drop Multiplexers

We will now concentrate on the case where the grooming factor is $g > 1$, for which we gave examples in Section 2.2. We first survey the main complexity and (in)approximability results. Then we see that for particular topologies and sets of connection requests, it is possible to obtain optimal constructions. We also consider the case where ADMs are placed a priori.

Let us first clarify the difference between *single-hop* and *multi-hop* (or *bifurcation allowed*) routing in this context. With single-hop routing, each request is routed through the same wavelength from its source to its destination. This is used for simple network topologies such as directed paths or rings, but not for general topologies where multi-hop routing is needed. When multi-hop routing is allowed, a request may be switched from one wavelength to another at intermediate nodes. This gives more flexibility for the traffic aggregation that is useful to optimize simultaneously the number of ADMs and wavelengths (see Section 2.5).

2.4.1 Complexity and Inapproximability Results

Determining the NP-completeness of the traffic grooming problem has been an open question for many years. It was first proved NP-complete on unidirectional rings in [21] using a reduction from the Bin Packing problem. Another proof was also mentioned in [112]. Later, the NP-completeness result has been refined.

More precisely, in [101] the traffic grooming problem is shown to be NP-complete *in the strong sense* for a given grooming factor $g \geq 2$, a network of directed path (or unidirectional ring) topology, a set of demands $I \subseteq K_N$, single-hop routing, and an unbounded number of colors (wavelengths). It is also shown to be NP-complete for rings and for paths for any fixed value $g \geq 2$, and when the number of colors is bounded.

The traffic grooming problem has also been proved NP-complete and hard to approximate in star networks in [74]. These results have been extended in [62] where a complete characterization of the traffic grooming problem complexity in star networks is given by providing optimal polynomial-time algorithms for $g \leq 2$ and proving the intractability of the problem for any fixed $g > 2$.

The first inapproximability result for traffic grooming with fixed values of the grooming factor g has been obtained in [2], thus answering affirmatively the conjecture of [23]. More precisely, it has been proved that traffic grooming on a unidirectional ring for fixed $g \geq 1$ and traffic grooming on a directed path for fixed $g \geq 2$ are APX-complete. That is, there is no polynomial-time approximation scheme (PTAS) with constant approximation factor for these problems, unless P = NP. Both results rely on the fact that finding the maximum number of edge-disjoint triangles in a graph (and more generally cycles of length $2g + 1$ in a graph of girth $2g + 1$) is APX-complete.

In particular, this implies that the traffic grooming problem is NP-complete in rings for fixed $g \geq 1$ and in paths for fixed $g \geq 2$ for an *unbounded* number of wavelengths, extending in this way the results of [101].

2.4.2 Approximation Results

The first approximation algorithm for the traffic grooming problem has been designed for the ring topology [71]. It is based on a greedy partition of the set of connection requests into trees of width at most g and has approximation ratio $\sqrt{g}$.

In [63] we present an approximation algorithm for the problem of minimizing the number of ADMs on a general network in the case where grooming is allowed. For every value of the grooming factor g the running time of the algorithm is polynomial in the input size. The approximation ratio of this algorithm for a wide variety of network topologies – including the ring topology – is shown to be $2\ln g + o(\ln g)$. In [62] the approximation ratio of the algorithm is shown to be $2\ln(\delta \cdot g) + o(\ln(\delta \cdot g))$ for any undirected tree having fixed node degree bound δ, and $2\ln g + o(\ln g)$ for unbounded degree directed trees.

As we have seen above, for general grooming factor g the best approximation algorithm [63] for the traffic grooming on a ring achieves an approximation factor of $\mathcal{O}(\log g)$, but its running time is exponential in g (that is, N^g). However, in practical applications such as SONET/SDH WDM rings, which are widely used as backbone optical networks [57], the grooming factor is equal to 3 or 4, typically when four 655 Mbit/s streams are aggregated into one 2.5 Gbit/s wavelength.

It is also important to find good approximation algorithms with running time polynomial in both N and g. Such approximation algorithm has been proposed in [2], where g is considered as part of the input. To the best of our knowledge, this is the first polynomial-time approximation algorithm for the traffic grooming problem with an approximation ratio which does not depend on g.

Theorem 2.1 (Amini et al. [2]). *There exists a polynomial-time approximation algorithm that approximates the traffic grooming problem on a ring within a factor of $\mathcal{O}(N^{1/3}\log^2 N)$ for any grooming factor $g \geq 1$.*

Theorem 2.2 (Amini et al. [2]). *There exists a polynomial-time approximation algorithm that approximates the traffic grooming problem on a path within a factor of $\mathcal{O}(N^{1/3}\log^2 N)$ for any grooming factor $g \geq 2$.*

Although the performance of this algorithm seems not to be very good at first sight, in fact it is conjectured in [2] that for the general instance of the problem it is not possible to get rid of a factor n^δ for some constant $\delta > 0$.

Finally, in [2] it is shown that the general scheme of the algorithm yields an $\mathcal{O}(\log^2 N)$-approximation if the request graph excludes a fixed graph as minor, for example, if R is planar or of bounded genus. The main theoretical contribution of this algorithm is to relate the traffic grooming problem to the dense k-subgraph problem [61] and the degree constrained subgraph problem [1].

2.4.3 Specific Constructions

For specific grooming factors, sets of requests and topologies, it is possible to give optimal constructions (assignment of requests to wavelengths that minimizes the number of ADMs). This is typically the case with all-to-all unitary traffic (one unitary request between each pair of nodes) where optimal constructions have been obtained on simple topologies for a specific grooming factor.

In unidirectional rings, all requests are routed clockwise. Therefore, it is possible to route requests (i, j) and (j, i) on the same wavelength at the cost of two ADMs and using $\frac{1}{g}$ of available bandwidth all along the ring. When the set of requests is symmetric, this is shown to be optimal [11]. Furthermore, in this case, the set of requests can be modeled by an undirected graph, each edge corresponding to a circle, and a subgraph B with g edges corresponds to a valid assignment of g circles to a wavelength. The number of nodes of B gives the number of ADMs to use on the corresponding wavelength. Therefore, the traffic grooming problem on a undirectional ring with symmetric traffic and grooming factor g can be modelled as the following partition problem.

Definition 2.1 (Traffic Grooming in Unidirectional Ring with Symmetric Traffic).

Input:	N nodes unidirectional cycle C_N, grooming factor g, and set of symmetric requests modeled by graph I.
Output:	Partition of I into subgraphs B_w, $1 \leq w \leq W$, such that $|B_w| \leq g$.
Objective:	Minimize $\sum_{w=1}^{W} |V(B_w)|$, and the optimum is denoted $A(C_N, I, g)$.

This problem is in general NP-complete. However, for the all-to-all unitary set of traffic requests, $I = K_N$, the complexity of the problem is unknown so far. Indeed, it is clearly a difficult combinatorial problem. Using tools of *Design Theory* [47], optimal constructions have been obtained for grooming factor $g = 3$ [5], $g = 4$ [11, 73], $g = 5$ [7], $g = 6$ [6], $g = 7$ [48], and $g \geq N(N-1)/6$ [11]. It has also been solved for practical values of N and g [9], that is, $N \leq 16$ and $g = 3, 4, 12, 16, 48, 64$.

When the physical topology is a directed path, the problem has only been solved for grooming factor $g = 2$, with all requests from left to right (transitive tournament, TT_N) [4]. As for traffic grooming on a unidirectional ring, the problem can be modeled as a graph partition problem. The main difficulty here is that the number of connections in each subgraph is subject to high variation since, for example, all requests $(i, i+1)$ may fit in the same subgraph (see [8] for the maximum value for any $g \geq 1$), and no suitable tools from graph or design theory have been developed so far. A formal definition of the problem for any valid set of connection requests is given in Definition 2.2, where $\text{load}(B_w, e)$ denotes the number of requests of B_w routed in the path through edge e.

Definition 2.2 (Traffic Grooming in Directed Path).

Input:	N nodes directed path P_N, grooming factor g, and set of requests I.
Output:	Partition of I into subgraphs B_w, $1 \leq w \leq W$, such that $\text{load}(B_w, e) \leq g$ for all $e \in P_N$.
Objective:	Minimize $\sum_{w=1}^{W} |V(B_w)|$, and the optimum is denoted $A(P_N, I, g)$.

Table 2.1 Congruence classes of N for some g for which optimal constructions are given

k	g	N
1	1	All values
2	3	$N \equiv 1,5 \mod 12$
3	6	$N \equiv 1,7 \mod 24$
4	10	$N \equiv 1,9 \mod 40$
5	15	$N \equiv 1,9 \mod 30$
6	21	$N \equiv 1,13 \mod 84$
7	28	$N \equiv 1,15 \mod 112$
8	36	$N \equiv 1,17 \mod 144$

Finally, when the physical topology is a bidirectional ring, the routing of the requests has to be taken into account since shortest path routing is not always optimal in general. However, it has been proved in [12] that symmetric shortest path routing allows us to obtain optimal solutions on bidirectional rings with all-to-all unitary traffic. The main results in this case are the following: optimal construction for the particular case $g = 1$ [13]; optimal construction when $g = 4, 8$ [49, 50]; optimal construction when $g = 3$ and $N \equiv 1, 5$ mod 12 [12] and when $g = k(k+1)/2$ for some congruence classes of N summarized in Table 2.1; and construction with approximation factor 12/11 when $g = 2$ [12].

2.4.4 A Priori Placement of the Equipment

In this section we study traffic grooming in unidirectional rings considering a wider range of requests than, for example, a complete graph. The idea is to place the ADMs in the nodes with limited knowledge of the graph of requests, for instance, knowledge of only its maximum degree. This model helps the network designer to take into account small traffic variations when deciding where to install ADMs, since in many situations one cannot expect to add or remove equipment at the nodes when the requests vary.

Namely, we consider the problem of placing the minimum number of ADMs in the nodes of a unidirectional ring in such a way that the network could support *any* request graph with maximum degree bounded by a constant Δ. Note that using this approach, as long as the degree of each node does not exceed Δ, the network can support a wide range of traffic demands without reconfiguring the equipment placed at the nodes. The problem can be formally stated as follows.

Definition 2.3 (Traffic Grooming in Unidirectional Rings with Bounded-Degree Symmetric Request Digraph).

Table 2.2 Values of $M(g,\Delta)$ found in [90]. The case $g=4$ and $\Delta=3$ is a conjectured value

$g \setminus \Delta$	1	2	3	4	5	6	...	Δ
$1,2$	1	2	3	4	5	6	...	Δ
3	1	2	3	≥ 3	≥ 4	≥ 4	...	$\geq \left\lceil \frac{2\Delta}{3} \right\rceil$
4	1	2	2??	≥ 3	≥ 4	≥ 4	...	$\geq \left\lceil \frac{5\Delta}{8} \right\rceil$
5	1	2	2	≥ 3	≥ 3	≥ 4	...	$\geq \left\lceil \frac{3\Delta}{5} \right\rceil$
$g \geq 5$	1	2	2	3	3	4	... $\geq$	$\left\lceil \frac{g+1}{g} \frac{\Delta}{2} \right\rceil$

Input: N nodes unidirectional cycle C_N, grooming factor g, and a maximum degree Δ.

Output: An assignment of $A(v)$ ADMs to each node $v \in V(C_N)$, in such a way that *for any request graph I* (each edge represents a pair of symmetric requests) with maximum degree at most Δ, there exists a partition of I into subgraphs B_λ, $1 \leq \lambda \leq \Lambda$, such that:

(i) $|E(B_\lambda)| \leq g$ for all λ; and
(ii)   each vertex $v \in V(C_N)$ appears in at most $A(v)$ subgraphs.

Objective: Minimize $\sum_{v \in V(C_N)} A(v)$, and the optimum is denoted $A(C_N,g,\Delta)$.

This problem has been studied in [90]. It solves the cases corresponding to $\Delta = 2$ (for all values of g) and $\Delta = 3$ (except for $g = 4$), and give upper and lower bounds for the general case. It also characterizes the function $A(C_N,g,\Delta)$, which turns out to be linear in N.

Lemma 2.1 ([90]). *The function $A(C_N,g,\Delta)$ is of the form $A(C_N,g,\Delta) = MN - \alpha$, where M and α are natural numbers depending only on g and Δ.*

A summary of the results of [90] is given in Table 2.2, where $M(g,\Delta)$ is the smallest integer such that the inequality $A(C_N,g,\Delta) \leq M(g,\Delta)N$ holds for any $N \geq 1$.

2.5 Multilayer Traffic Grooming for General Networks

In previous sections we have discussed various aspects of traffic grooming for ring and tree networks. In this section we will discuss the case of more general network topologies, typically referred to as *mesh networks*. First we give an overview of different architectures of practical interest; then we give a survey of different graph models used with an ILP formulation and show examples of what can these models be used for.

2.5.1 Multilayer Mesh Networks

If there are multiple network layers "one over the other," we refer to this structure as "Multilayer" network. It is also referred to as the vertical structure of networks, in contrast to the horizontal, where multiple domains are mutually interconnected. These network layers are not the ISO-OSI layers, where each layer defines some network functionality, but layers that can each provide certain connections or virtual connections and that can be established using the same or different network technologies.

Examples where the same network technology is used are the old FDM (Frequency Division Multiplexed) systems, different ATM (Asynchronous Transfer Mode) networks with two layers, namely VP and VC layers, and the MPLS (Multi-protocol Label Switching) networks where practically any number of LSPs (Label Switched Paths) can be established, where the lower-layer paths are considered as links in the upper-layer. In this case, the upper-layer paths share these lower-layer paths, i.e., they are encapsulated or embedded into these paths.

Examples where different technologies are used are

- PDH over SDH
- IP over PoS/MAPOS over SDH over WDM
- IP over ATM/MPLS over SDH over WDM
- IP over GFP over SDH over OTN over WDM
- IP over PPP over Ethernet over ATM-AAL5 over SDH over OTN ...

A multilayer network consists in general of interconnected multilayer and single-layer nodes. The single-layer nodes can be at any network layer, while multilayer nodes are those that are attached to two or more layers and/or perform the switching at two or more layers.

There are two general specifications of such multilayer architectures one referred to as GMPLS (Generalized Multi-protocol Label Switching) by the IETF [86] and the other ASTN (Automatic Switched Transport Network) by the ITU-T [76].

The IETF GMPLS framework [98] defines the following layers, this time according to the switching capability, i.e., a layer can be established by different networking technologies:

- PSC (Packet Switching Capable, e.g., IP)
- L2SC (Layer 2 Switching Capable, e.g., GbEth)
- TSC (TDM Switching Capable, e.g., SDH VC-4-4c)
- λSC (Wavelength Switching Capable)
- WBSC (WaveBand Switching Capable)
- FSC (Fiber Switching Capable)

Typically not all these layers are represented in a network, but rather only two or three of them. Having multiple layers has both advantages and disadvantages. The advantages are that the services can access finer bandwidth granularity and some additional features of upper-layers only, i.e., for a small ratio of traffic only. The drawbacks are that some functionality is multiplied across layers and that the

complexity of operating multilayer networks is much higher than that of operating certain layers separately.

This layered vertical structure is valid for the data plane (DP), i.e., the network that carries the user information. However, for configuring and operating such a network we need a management and a control plane (MP and CP respectively).

If the DP layers of this vertical structure are run by different operators or providers, then they must communicate with each other to exchange information necessary for routing and other purposes. This vertical communication between MP and CP layers is referred to as Interconnection, and there are three defined *Interconnection Models*: (1) Overlay, (2) Augmented, and (3) Peer model [98].

The *Overlay* model is a client-server model where the upper (client) layer always adapts to the lower (server) layer. In the case of the *Peer* model, all necessary information is interchanged between the layers, and they may act together, e.g., in routing a demand. The *Augmented* (or hybrid) model is somewhere in between the Overlay and Peer models.

The DP layers in a node can be controlled either each by its own CP instance that communicates with other layers of that node, or by a single CP instance that controls all the DP layers of that node.

The latter case is feasible only if all the DP layers are run by a single operator or provider, since there is no need for communication interfaces between the layers. Therefore, a single unified integrated CP can be used for all the layers instead of the interconnection, the so-called Integrated Model. The forwarding units of all the layers of the data plane are connected to a single control plane unit.

Similarly, if such a multilayer network has layers or some parts of certain layers built of interconnected elements of a unique networking technology, or, rather switching capability, then the set of these elements is defined by the CCAMP WG of IETF as a Region. A network having multiple different regions is referred to as a Multi-region network [93, 104].

2.5.2 On Grooming in Multilayer Mesh Networks

In switched multilayer transport networks (e.g., ASTN/GMPLS) the traffic demands have typically bandwidth of orders of magnitude lower than the capacity of wavelength links (λ-links). Therefore, it is not worth assigning exclusive end-to-end wavelength paths (λ-paths) to these demands, i.e., sub-λ granularity is required. Furthermore, the number of wavelengths per fiber is limited and costly. To increase the throughput of a network with a limited number of wavelengths per fiber, traffic grooming capability is required in certain nodes.

Here we assume two layers only, i.e., a Wavelength Routing Dense Wavelength Division Multiplexing (WR-DWDM) Network and one layer built over it. In the WR-DWDM layer, a λ-path connects two physically adjacent or distant nodes. These two physical nodes will seem adjacent for the upper layer built over it. More generally, we can consider this two-layer approach as two layers of a 4–6 layer GM-

PLS/ASTN architecture [98]. However, not only is the framing and layering structure of interest, but the control plane proposed in the GMPLS/ASTN framework is as well.

This upper layer is an "electronic" one, i.e., it can perform multiplexing different traffic streams into a single λ-path via simultaneous time and space switching. Similarly, it can demultiplex different traffic streams of a single λ-path. Furthermore, it can perform re-multiplexing as well: Some of the demultiplexed demands can be again multiplexed into some other λ-paths and handled together along them. This is often referred to as (traffic) grooming [27]. The electronic layer is required for multiplexing packets coming from different ports (asynchronous time division multiplexing).

This upper electronic layer can be a classical or "next generation" SDH/SONET, MPLS, ATM, GbE, or 10 GbE, or it can be based on any other technology. However, in all cases the network carries mostly IP traffic. The only requirement is that it must be identical for all traffic streams that have to be demultiplexed, and then multiplexed again, since we cannot multiplex, e.g., ATM cells with Ethernet frames directly.

2.5.3 Graph Models for Multilayer Grooming

Optical metro and particularly core networks consist of multiple layers, where multiple different networking technologies are stacked one over the other. For simplicity, here we assume two layers only, e.g., an IP/MPLS layer over an DWDM layer, both controlled jointly by either one vertically peer-interconnected or one vertically integrated GMPLS control plane.

To better utilize network resources, smaller, upper-layer traffic streams are multiplexed ("groomed") into higher capacity wavelength paths in a distributed way throughout the network.

In this section we give an overview of known graph models as well as propose some new graph models that all allow both static and dynamic grooming while performing design, dimensioning, configuration, routing, multicasting, traffic engineering, and resilience functions.

2.5.3.1 Grooming and Wavelength Assignment for Static Routing

The aim of the Grooming and Wavelength Assignment for Static Routing problem (or, for short, Static Grooming problem) is to find a static configuration of the virtual (logical) topology, and to assign the upper layer demands to this topology. It is assumed that the lower network topology, the number of wavelengths per link, the capacity of these links, and the traffic matrix is given.

The simplest case of static grooming is when the routing is given, and the routes of certain demands are to be bundled (groomed together) in certain parts of the network and assigned to a wavelength.

In [24, 31, 33], a simple model and various heuristic algorithms based on simulated annealing, threshold accepting, and tabu search, as well as a genetic algorithm, are proposed and evaluated. The idea of the model is that each part of a route along each link can be assigned to any wavelength if that wavelength has enough free capacity to accommodate the considered demand. The objective is to have as few groomings and wavelength conversions as possible. The elementary heuristic step is to try out different combinations of assigning a segment of a path to different wavelength links, where the improvements are accepted with higher probability.

The first model for static grooming where the routing was not given in advance but performed simultaneously with grooming and wavelength assignment was proposed in [32]. Later, a method based on ILP formulation for optimal configuration was proposed in [43], and due to complexity simple heuristic methods using the same graph model were proposed in [44].

The wavelength graph model proposed in [32] is as follows. For each fiber link $l = (u, v)$ with Λ wavelengths from u to v we create Λ arcs, one per wavelength, from vertex $u_{l,\lambda}$ to vertex $v_{l,\lambda}$, $1 \leq \lambda \leq \Lambda$. Thus, node u with L_{in} incoming links and L_{out} outgoing links is associated with vertices $u_{l_{in},\lambda}$ and $u_{l_{out},\lambda}$, $1 \leq l_{in} \leq L_{in}$ and $1 \leq l_{out} \leq L_{out}$, and a bipartite digraph from vertices $\left\{u_{l_{in},\lambda}\right\}$ to vertices $\left\{u_{l_{in},\lambda}\right\}$ modeled possible interconnections in network node u. This bipartite digraph will be complete if it is possible to switch from any wavelength to any other.

The ILP formulation [43] uses the proposed graph model, and finds the minimal cost multi-commodity flow over the graph according to the traffic matrix and the costs assigned to the edges of the graph. However, the ILP can be solved optimally for very small instances only.

Heuristics based on the decomposition into as many shortest path searches as nonzero elements in the traffic matrix were proposed in [44]. Here, empirical weighting of edges has been also proposed to improve the quality of results. In contrast to the ILP that gives exact globally optimal results (for very small network instances), this approach is an approximation only. It is however easily scalable to very large networks, since it is based on Dijkstra's algorithm.

In [110] a heuristic method based on decomposition and iterations has been proposed that also contains elements of simulated annealing and tabu search. The idea was that an element of a traffic matrix is a demand that goes from node a to node c; however, instead of setting up an end-to-end wavelength path we can use two shorter lightpaths via an intermediate node b. Then it corresponds to a new traffic matrix, where elements a to b and b to c are increased by the bandwidth of demand a–c while this a–c entry is decreased by its bandwidth (typically to 0). In this case a simpler graph model was used [22] that originally did not support grooming but only wavelength routing and assignment in a single-layer network; however, grooming was handled through the traffic matrix transformations.

The use of Integer Linear Programming ensures that the solution is the global optimum in terms of the given objective function. However, as the problem to be

solved becomes more complex, and as the network size increases, ILP can become intractable (in particular for NP-hard problems). Still, it is worth using it as a reference, at least for smaller networks. As computing capacity grows, particularly due to the parallelism of supercomputers, GRIDs, and clusters this will also become a viable solution.

As already mentioned, in [43] an ILP formulation for the wavelength graph has been given. In [25, 26] the formulation has been extended for undirected graphs as well, with protection either at the upper or at the lower layer.

2.5.3.2 Network Dimensioning and Grooming Node Placement

For a two-layer network, both the layers and the interconnection points between the two layers must be dimensioned properly. However, due to the interactions of the layers, all three must be dimensioned simultaneously, leading to high complexity.

In a network it is not necessary to equip all the nodes with grooming capability. Furthermore, since the O/E (Opto-Electronic) and E/O (Electro-Optical) converters are very expensive, their numbers should be properly determined to reduce costs while maintaining proper operation of the network. In [94] three methods are proposed for deciding which nodes should perform grooming, and to dimension their grooming capacity. The three methods are a greedy approach, a vertex-cover-based approach, and a heuristic approach that sorts the nodes according to their eligibility for accommodating grooming capability. The three methods have similar performance. In all cases the wavelength graph has been used.

In [96] a simulation-based iterative heuristic method has been proposed. Its idea is that simulations are run for infinite grooming capability in all nodes, and statistics (probability density functions, or pdfs) of the resource usage are compiled. Based on these pdfs it is decided in which nodes to keep the grooming capability and how much to reduce it. Then simulations are repeated and the whole process continued iteratively.

In [95] the optimization objective was extended to optimise not only the grooming capability, but simultaneously the number of wavelengths to be used per fiber as well.

2.5.3.3 Grooming for Dynamic Routing

"Grooming for dynamic routing" or "dynamic grooming" means, that in an operational network the new demands arrive while the demands already routed get terminated sooner or later, i.e., the network changes dynamically. In contrast to static grooming this is a less complex problem, since a single demand has to be routed at a time and groomed together with some existing demands; however, it is hard to say what is the globally optimal long-term strategy.

Here we discuss some related papers.

In [109] the information multi-domain multilayer (MD-ML) influence of delay of advertisements and inaccuracies due to the topology and link state aggregation is studied in an MD-ML network. The wavelength graph model has been used; however, this information is available only within the domains. Over domain boundaries a simplified aggregated graph is advertised.

In [38, 39] the advantages and drawbacks are investigated of having both layers switched according to user demands compared to the case where the WDM system is fixed, and only rarely reconfigured, while over this virtual topology the demands are dynamically routed. Here, an enhanced version of the wavelength graph is used that we refer to as the Grooming Graph or the Fragment Graph, where a wavelength path can be cut into two or more shorter pieces and two or more shorter wavelength paths can be concatenated into a longer one to reduce the load of the electronic layer.

Finally, [79] gives an overview of routing demands of different traffic parameters (e.g., very different bandwidths) over multilayer multi-domain networks.

2.5.3.4 VPN, oVPN, VPλN and VON, oVON, VOλN

Virtual Private Networks (VPNs), as well as Virtual Overlay Networks (VONs), are virtual networks set over real physical networks by separating a part of physical resources, e.g., link and switching capacities. We will refer to these jointly as VNs (Virtual Networks). When multilayer networks are considered, two main options can be differentiated: First, when the virtual topology provided by the lower layer is shared among the VPNs or VONs of the upper layer; second, when the VNs are the virtual topology, i.e., the wavelength paths are the links of the VNs.

In [85] multi-fiber WDM networks are considered. In this paper full wavelength conversion capability is assumed in all nodes; therefore, no wavelength continuity constraint has to be obeyed, but only as many parallel links as the product of the number of existing fibers and wavelengths. Heuristics based on decomposition and Suurballe's shortest pair of paths algorithms (cf. Section 1.5.2.1) are used to determine the best failure-resistant VPNs either demand-by-demand or VPN-by-VPN.

In [28, 87] open VPNs (oVPNs) are optimized by using ILPs while obeying the wavelength continuity constraint. ILP formulations for the cases without and with protection are given. For the case with protection two sub-cases are defined: One with external protection, where the network provider is supposed to protect the VPN, the other with internal protection, when the VPN is configured in such a way that if any link or node fails, the resources of the VPN are used for protection. Finally, in [88] more physical limitations are considered for setting up wavelength paths.

2.5.3.5 Grooming for Multicast Traffic

Services like TV or video distribution can be more efficiently provided using point-to-multipoint tree structures rather than many point-to-point connections. These ser-

vices have become increasingly more popular, and the bandwidth used by these services has also grown, i.e., unlike standard definition digital video, high-definition video is already streamed.

If not a single channel, but rather a bundle of programs is streamed simultaneously, this bandwidth may achieve or even exhaust the capacity of a single wavelength channel. Therefore, performing the multicast at the optical layer via a splitter can be a much cheaper solution than loading the electronic layer with all the multicasting.

In [107] multicast trees are obtained by ILP. Breadth and depth constraints are obeyed, and it has been evaluated how many ports and how many wavelengths (resources in general) are needed for electronic and optical signal branching and how many for unicast as a reference.

The wavelength graph model has been used again; however, it had to be modified to allow branching of the optical signal, which was not allowed for unicast demands.

In [97] methods for periodical reconfiguration of multicast trees has been proposed for two-layer grooming-capable networks. Multicast trees (light trees) change dynamically in time due to the changing of multicast endpoints, which causes degradation of the tree. A significant amount of network resources can be saved by regular reconfiguration. The benefit of reconfiguration is investigated for different routing algorithms and reconfiguration periods.

In [45] various restoration mechanisms for multicast trees are considered.

2.5.3.6 Grooming and Resilience

In two-layer grooming-capable networks the demands can be routed over either the upper or the lower layers, or even using both layers. The same holds for routing the protection paths of these demands. For dedicated protection only an SRG (Shared Risk Group) disjoint path is to be sought; however, for the case of shared path protection this becomes more complex. Namely, not only the capacity is shared, but also are the O/E and E/O conversion ports as well as the wavelength paths.

In [25, 26] an ILP formulation for different Dedicated Protection Schemes is presented, while in [68] a decomposition-based heuristic method has been proposed for the same purpose. In [66] different methods based on running Dijkstra's algorithm twice or Suurballe's algorithm for static grooming are presented (cf. Section 1.5.2.1). In [67] the difference is that *dynamic* grooming is assumed, i.e., demands arrive one by one and are both routed and protected instantly.

In [34] shared protection is proposed and fairness issues in terms of dependence on bandwidth and distances are investigated.

In [41, 42] a new version of the wavelength graph model has been introduced that allows not only setting up and tearing down lightpaths, but also fragmenting and defragmenting them. The idea is that if there are no free wavelength paths in a node, then an existing wavelength path can be cut ("fragmented") and the new demand is added or dropped at that point. If there are two consequent wavelength paths carrying the same demand or demands, these can be concatenated, i.e., "defragmented."

Therefore we refer to this model as "Fragment Graph." Here, the routing of working and shared protection paths are considered simultaneously.

In contrast to the previous papers in [78], an Ethernet over WDM overlay is considered, where we compare different configurations of the wavelength path system of the WDM layer and optimally set up MSTP (Multiple Spanning Tree Protocol) trees of the Ethernet layer.

All the methods discussed in this subsection use the wavelength graph model except the last one, which assumes an overlay model, so a simpler graph is sufficient.

2.5.3.7 Traffic Engineering for Traffic Grooming

The simplest definition of Traffic Engineering (TE) is to "put the traffic where enough resources are available." It can be considered as an improved adaptive routing. The adaptivity can be achieved in two ways. First, by setting edge weights in our graph to avoid congestions and higher blocking before they occur ("a priori"). Second, by applying wavelength path fragmentation and defragmentation as already explained in Subsection 2.5.3.6 to resolve existing congestions for newly arriving demands ("a posteriori"). Here we give a short overview of MLTE-(Multilayer Traffic Engineering)-related papers.

A general overview of TE in GMPLS controlled multilayer networks is presented in [111].

Several adaptive edge metrics (weights) for MLTE have been proposed and compared in [99], using a simpler graph model than in [80]. Then, adaptive fragmentation and defragmentation of wavelength paths is proposed in [35–37] and compared to the case with no fragmentation or defragmentation and to the case with OXCs only (i.e., no grooming capability). Next, [29] gives an overview of achievements of Routing TE and resilience in Heterogeneous-GMPLS-controlled networks, while [77] presents experimental results from European testbeds.

Finally, we discuss three papers [30, 40, 83] that perform joint "a priori" and "a posteriori" Traffic Engineering. The idea is that although the fragment graph (FG) is being used for performing "a posteriori" TE, and the edge weights of the FG are as follows. Assuming that roughly no more than half of the demands can be terminated and O/E – E/O converted, the load should be balanced accordingly. i.e., if there are few demands routed over the network and therefore few wavelengths are used, the longer wavelength paths with less electronic processing (grooming) are made cheaper. However, if more wavelengths start to be used, but the capacity of these wavelengths is not well utilized the cost of grooming will decrease leading to shorter paths over more and shorter fragments.

2.5.3.8 Cross-layer Optimization: Considering Physical Impairments While Routing

Often, in networks it is not enough to consider the available resources, but it is also necessary to consider the impairments that affect the signal quality at the physical layer and cause increased Bit Error Rate for services. This is a kind of cross-layer optimization, where the services are optimized with physical layer constraints.

The first use of grooming to repair the impaired signal was presented in [88] where such VPNs were configured, where the signal quality was satisfactory since the physical impairments were considered. In [120] the results were extended for routing in general. [107, 119] present deeper results on the same topic, while [46] gives an overview of the problem, and proposes an additional method for improving the signal quality by increasing the power level of signals that have to go far while decreasing the power levels of signals that go to closer destinations in order to avoid the harmful effect of nonlinear distortions.

2.6 Conclusion

The objective of this chapter was to present an overview of traffic grooming in connection-oriented networks (mainly in WDM networks) and the wide variety of mathematical tools used to address this issue. Traffic grooming refers to techniques used for an efficient sharing of the bandwitdh offers by, e.g., a wavelength, using Time Division Multiplexing. It is usually associated with the routing of the requests and the survivability issue in single or multiple failure scenarios. Furthermore, traffic requests might be uni- or multicast, the traffic pattern may evolve with time, and the network could be multilayer. Therefore, traffic grooming is only part of the concerns addressed when designing and optimizing a network. But even when restricted to simple physical topologies (unidirectional path or ring) where the routing is fixed and with small grooming factor, the traffic grooming problem is difficult to solve and to approximate. Also, when all aspects have to be taken into account (traffic grooming, routing, survivability, and so on), problems to solve are so difficult that exact solutions are usually no longer expected, and it is essential to develop efficient heuristic algorithms. Some of them were presented in Section 2.5. Chapter 3 presents the state-of-the-art regarding exact approaches for this problem.

In this research area, several important questions are still open and further research are needed. In particular, when optimizing only the number of ADMs in SONET/SDH networks, practical values of the grooming factor are 3 and 4, but this is reapeated several time from the slower 55 Mbit/s streams to the current 10 Gbit/s wavelengths. So, it is important to develop efficient optimization tools for grooming factors 3 and 4, but also to consider hierarchical problems in which unitary requests are combined by four into streams that are themselves combined by four, and so on.

In the general context, where traffic grooming is associated with routing and survivability issues, existing heuristic algorithms provide upper bounds without guar-

antee on the quality of the solution. Furthermore, the size of practical problems is too huge for existing mathematical tools. Therefore, research effort has to be put into the development of new mathematical tools allowing us to address large instances and to obtain optimal or near-optimal solutions.

References

1. Amini, O., Peleg, D., Perennes, S., Sau, I., Saurabh, S.: Degree-constrained subgraph problems: Hardness and approximation results. In: 6th Workshop on Approximation and Online Algorithms (WAOA), *Lecture Notes in Computer Science*, vol. 5426, pp. 29–42. Springer (2009)
2. Amini, O., Pérennes, S., Sau Valls, I.: Hardness and approximation of traffic grooming. In: The 18th International Symposium on Algorithms and Computation (ISAAC 2007). Sendai, Japan (2007)
3. Beauquier, B., Bermond, J. C., Gargano, L., Hell, P., Perennes, S., Vaccaro, U.: Graph problems arising from wavelength–routing in all–optical networks. In: Proc. Conference WOCS97, Geneva,April 1997 (1997)
4. Bermond, J. C., Braud, L., Coudert, D.: Traffic grooming on the path. Theoretical Computer Science **384**(2-3), 139–151 (2007)
5. Bermond, J. C., Ceroi, S.: Minimizing SONET ADMs in unidirectional WDM ring with grooming ratio 3. Networks **41**(2), 83–86 (2003)
6. Bermond, J. C., Colbourn, C., Coudert, D., Ge, G., Ling, A., Muñoz, X.: Traffic grooming in unidirectional WDM rings with grooming ratio C=6. SIAM Journal on Discrete Mathematics **19**(2), 523–542 (2005). DOI 10.1137/S0895480104444314
7. Bermond, J. C., Colbourn, C., Ling, A., Yu, M. L.: Grooming in unidirectional rings: $k_4 - e$ designs. Discrete Mathematics, Lindner's Volume **284**(1-3), 57–62 (2004)
8. Bermond, J. C., Cosnard, M., Coudert, D., Perennes, S.: Optimal solution of the maximum all request path grooming problem. In: Advanced International Conference on Telecommunications (AICT). IEEE (2006)
9. Bermond, J. C., Coudert, D.: Traffic grooming in unidirectional WDM ring networks using design theory. In: IEEE ICC, vol. 2, pp. 1402–1406. Anchorage, Alaska (2003). ON07-3
10. Bermond, J. C., Coudert, D.: Handbook of Combinatorial Designs (2nd edition), *Discrete Mathematics and Applications*, vol. 42, chap. VI.27, Grooming, pp. 494–496. Chapman & Hall- CRC Press, editors C. J. Colbourn and J. H. Dinitz (2006)
11. Bermond, J. C., Coudert, D., Muñoz, X.: Traffic grooming in unidirectional WDM ring networks: The all-to-all unitary case. In: The 7th IFIP Working Conference on Optical Network Design & Modelling – ONDM, pp. 1135–1153. Budapest, Hongrie (2003)
12. Bermond, J. C., Coudert, D., Muñoz, X., Sau Valls, I.: Traffic grooming in bidirectional WDM ring networks. In: IEEE-LEOS ICTON / COST 293 GRAAL, vol. 3, pp. 19–22 (2006)
13. Bermond, J. C., Coudert, D., Yu, M. L.: On DRC-Covering of K_n by cycles. Journal of Combinatorial Designs **11**(2), 100–112 (2003)
14. Bilò, V., Flammini, M., Moscardelli, L.: On nash equilibria in non-cooperative all-optical networks. In: STACS, *Lecture Notes in Computer Science*, vol. 3404, pp. 448–459. Springer (2005)

15. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
16. Brehon, Y., Kofman, D.: Bus-label switched paths, an approach to reduce the cost of multilayer networks. In: IEEE International Conference on Communication (ICC), vol. 5, pp. 2401 – 2406. Istanbul (2006). DOI 10.1109/ICC.2006.255129
17. Brehon, Y., Kofman, D., Pioro, M., Diallo, M.: Optimal virtual topology design using bus-label switched paths. IEEE Journal on Selected Areas in Communications **25**(5), 1001 – 1010 (2007). DOI 10.1109/JSAC.2007.364261
18. Cao, X., Anand, V., Qiao, C.: Waveband switching for dynamic traffic demands in multigranular optical networks. IEEE/ACM Transactions on Networking (2007). DOI 10.1109/TNET.2007.896234
19. Chen, L. W., Modiano, E., Saengudomlert, P.: Optimal waveband switching in WDM networks. In: IEEE International Conference on Communication (ICC), pp. 1604 – 1608. Paris, France (2004). DOI 10.1109/ICC.2004.1312781
20. Chen, L. W., Modiano, E., Saengudomlert, P.: Uniform versus non-uniform band switching in WDM networks. Computer Networks **50**(2), 149–167 (2006). DOI doi:10.1016/j.comnet.2005.05.017
21. Chiu, A. L., Modiano, E. H.: Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks. IEEE/OSA Journal of Lightwave Technology **18**(1), 2–12 (2000)
22. Chlamtac, I., Faragó, A., Zhang, T.: Lightpath (wavelength) routing in large WDM networks. IEEE Journal on Select. Areas Commun. **14**, 909–913 (1996)
23. Chow, T. Y., Lin, P. J.: The Ring Grooming Problem. Networks **44**, 194–202 (2004)
24. Cinkler, T.: Heuristic algorithms for configuration of the ATM-layer over optical networks. In: ICC'97, IEEE International Conference on Communications, pp. 1164–1168. Montreal, Canada (1997)
25. Cinkler, T.: ILP formulation of grooming over wavelength routing with protection. In: ONDM 2001, IFIP ONDM 5th Conference on Optical Network Design and Modeling, pp. 25–48. Kluwer, Vienna (2001)
26. Cinkler, T.: ILP formulation of grooming over wavelength routing with protection. In: A. Jukan (ed.) Towards an Optical Internet, New Visions in Optical Network Design and Modelling, pp. 25–48. Kluwer (2002)
27. Cinkler, T.: Traffic- and λ-grooming. IEEE Network **17**(2), 16–21 (2003)
28. Cinkler, T.: Configuring traffic grooming VPλNs. Special Issue of the Photonic Network Communications on Optical Virtual Private Networks (oVPNs) **7**(3), 239–253 (2004)
29. Cinkler, T.: Routing, TE and resilience in heterogeneous GMPLS networks. In: MPLS/GMPLS Workshop. Girona, Spain (2006)
30. Cinkler, T.: On 'a priori' and 'a posteriori' multi-layer traffic engineering schemes. In: IEEE ICC 2007, IEEE International Conference on Communications – WorkShop on Traffic Engineering in Next-Generation IP Networks. Glasgow, Scotland (2007)
31. Cinkler, T., Ast, L., Faragó, A., Henk, T.: Configuration of the ATM-layer over optical networks. In: Fifth International Conference on Telecommunication Systems Modelling and Analysis. Nashville, TN (1997)
32. Cinkler, T., Castro, R., Johansson, S.: Configuration and re-configuration of WDM networks. In: NOC'98, European Conference on Networks & Optical Communications, pp. 6–13. Manchester, UK (1998)
33. Cinkler, T., Cinkler, K., Trón, T., Halász, E.: On almost-all-optical ATM networks. In: Eighth IEEE Workshop on Local and Metropolitan Area Networks. Potsdam (1996)
34. Cinkler, T., Gáspár, C.: Fairness issues of routing with grooming and shared protection. In: ONDM 2004, 8th Conference on Optical Network Design and Modelling, pp. 665–684. Ghent, Belgium (2004)
35. Cinkler, T., Geleji, G., Asztalos, M., Hegyi, P., Kern, A., Szigeti, J.: Lambda-path fragmentation and de-fragmentation through dynamic grooming. In: IEEE ICTON 2005, 7th International Conference on Transparent Optical Networks. Barcelona, Spain (2005)

36. Cinkler, T., Hegyi, P., Asztalos, M., Geleji, G., Szigeti, J., Kern, A.: Multi-layer traffic engineering through adaptive λ-path fragmentation and de-fragmentation. In: F. Boavida, T. Plagemann, B. Stiller, C. Westphal, E. Monteiro (eds.) Networking, *Lecture Notes in Computer Science*, vol. 3976 / 2006, pp. 715–726. Springer, Berlin/Heidelberg (2006)

37. Cinkler, T., Hegyi, P., Asztalos, M., Geleji, G., Szigeti, J., Kern, A.: Multi-layer traffic engineering through adaptive lambda-path fragmentation and de-fragmentation: The "grooming-graph" and the "shadow-capacities". In: IFIP Networking. Coimbra, Portugal (2006)

38. Cinkler, T., Hegyi, P., Geleji, G., Szigeti, J.: Configured vs. switched underlying wavelength system for traffic engineering with grooming. In: TelFor 2006, 14th Telecommunications Forum. Belgrade, Serbia (2006)

39. Cinkler, T., Hegyi, P., Geleji, G., Szigeti, J.: Multi-layer traffic engineering: Should only the upper-most layer be switched? In: ONDM 2006. Copenhagen, Danemark (2006)

40. Cinkler, T., Hegyi, P., Geleji, G., Szigeti, J.: Fairness issues of AMLTE: Adaptive multi-layer traffic engineering with grooming. In: ICTON 2007, 9th International Conference on Transparent Optical Networks. Rome, Italy (2007)

41. Cinkler, T., Hegyi, P., Geleji, G., Szigeti, J.: Protection for adaptive multi-layer traffic engineering. In: DRCN 2007, 6th International Workshop on the Design of Reliable Communication Networks. La Rochelle, France (2007)

42. Cinkler, T., Hegyi, P., Geleji, G., Szigeti, J.: Adaptive multi-layer traffic engineering with shared risk group protection. In: ICC2008, IEEE International Conference on Communications, Optical Networks and Systems Symposium. Beijing, China (2008)

43. Cinkler, T., Larsen, C. P.: Integer linear programming (ILP) formulation of the optimal light-path configuration problem in wavelength-routing WDM networks. In: ONDM'99, Third Working Conference on Optical Network Design and Modelling, IFIP TC6/WG 6.10 (Task Group on Photonic Communication Networks). Paris, France (1999)

44. Cinkler, T., Marx, D., Larsen, C. P., Fogaras, D.: Heuristic algorithms for joint configuration of the optical and electrical layer in multi-hop wavelength routing networks. In: IEEE INFOCOM 2000, pp. 1000–1009. Tel Aviv (2000)

45. Cinkler, T., Perényi, M., Soproni, P.: Restoration of multi-cast trees in optical-beared grooming-capable two-layer networks. In: ICTON 2008, 10th Aniversary International Conference on Transparent Optical Networks, pp. 14–18. Athens, Greece (2008)

46. Cinkler, T., Zsigmond, S., Perényi, M.: Traffic grooming and power level tuning for physical impairment constrained routing. In: ICTON 2008, 10th Aniversary International Conference on Transparent Optical Networks, pp. 162–165. Athens, Greece (2008)

47. Colbourn, C., Dinitz, J. (eds.): Handbook of Combinatorial Designs, 2 edn. Chapman & Hall/CRC (2006)

48. Colbourn, C., Fu, H. L., Ge, G., Ling, A., Lu, H. C.: Minimizing SONET ADMs in unidirectional WDM rings with grooming ratio seven. SIAM J. Discrete Mathematics **23**(1), 109–122 (2008)

49. Colbourn, C., Ling, A.: Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks. Discrete Mathematics **261**(1-3), 141–156 (2003)

50. Colbourn, C., Wan, P. J.: Minimzing drop cost for SONET/WDM networks with $\frac{1}{8}$ wavelength requirements. Networks **37**(2), 107–116 (2001)

51. Crouser, R., Rice, B., Sampson, A.: On-line distributed traffic grooming. In: IEEE International Conference on Communication (ICC), pp. 5239 – 5246. Beijing (2008). DOI 10.1109/ICC.2008.984

52. Călinescu, G., Frieder, O., Wan, P. J.: Minimizing electronic line terminals for automatic ring protection in general WDM optical networks. IEEE Journal of Selected Area on Communications **20**(1), 183–189 (2002)

53. Călinescu, G., Wan, P. J.: Splitable traffic partition in WDM/SONET rings to minimize SONET ADMs. Theoretical Computer Science **276**(1-2), 33–50 (2002)

54. Di Giannantonio, S., Flammini, M., Monaco, G., Moscardelli, L., Shalom, M., Zaks, S.: Selfishness, collusion and power of local search for the ADMs minimization problem. In: S. B.. Heidelberg (ed.) WINE, Lecture Notes in Computer Science (2007)

55. Dutta, R., Kamal, A. E., Rouskas, G. N. (eds.): Traffic Grooming for Optical Networks: Foundations, Techniques and Frontiers. Optical Networks. Springer (2008)

56. Dutta, R., Rouskas, N.: A survey of virtual topology design algorithms for wavelength routed optical networks. Optical Networks **1**(1), 73–89 (2000)

57. Dutta, R., Rouskas, N.: Traffic grooming in WDM networks: Past and future. IEEE Network **16**(6), 46–56 (2002)

58. Eilam, T., Moran, S., Zaks, S.: Lightpath arrangement in survivable rings to minimize the switching cost. IEEE Journal of Selected Area on Communications **20**(1), 172–182 (2002)

59. Epstein, L., Levin, A.: Better bounds for minimizing SONET ADMs. In: S. B.. Heidelberg (ed.) 2nd Workshop on Approximation and Online Algorithms (WAOA), *Lecture Notes in Computer Science*, vol. 3351, pp. 281–294. Bergen, Norway (2004). DOI 10.1007/b106130

60. Epstein, L., Levin, A.: SONET ADMs minimization with divisible paths. In: S. B.. Heidelberg (ed.) 3rd International Workshop on Approximation and Online Algorithms – WAOA, *Lecture Notes in Computer Science*, vol. 3879, pp. 119–132 (2005). DOI 10.1007/11671411_10

61. Feige, U., Peleg, D., Kortsarz, G.: The Dense *k*-Subgraph Problem. Algorithmica **29**(3), 410–421 (2001)

62. Flammini, M., Monaco, G., Moscardelli, L., Shalom, M., Zaks, S.: Approximating the traffic grooming problem in tree and star networks. In: WG, *Lecture Notes in Computer Science*, vol. 4271, pp. 147–158. Springer (2006)

63. Flammini, M., Moscardelli, L., Shalom, M., Zaks, S.: Approximating the traffic grooming problem. Journal of Discrete Algorithms **6**(3), 472–479 (2008)

64. Flammini, M., Shalom, M., Zaks, S.: On minimizing the number of ADMs – tight bounds for an algorithm without preprocessing. Journal of Parallel and Distributed Computing **67**(4), 448–455 (2007). Also in Proceedings of the 3rd Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN), Chester, United Kingdom, July 2006

65. Flammini, M., Shalom, M., Zaks, S.: On minimizing the number of ADMs in a general topology optical network. Discrete Applied Mathematics **157**, 2701–2717 (2009). Also in Proceedings of 20th International Workshop on Distributed Algorithms (DISC), Stockholm, Sweden, September 2006

66. Fülöp, L., Cinkler, T.: Algorithms for grooming over wavelength routing with protection. In: DRCN 3rd Conference on Design of Reliable Communication Networks, pp. 160–167. Budapest (2001)

67. Gáspár, C., Makács, G., Cinkler, T., Tapolcai, J.: Wavelength routing with grooming and protection. In: IFIP ONDM 2003, 7th Conference on Optical Network Design and Modelling. Budapest, Hungary (2003)

68. Gáspár, C., Szentes, S., Tapolcai, J., Cinkler, T.: Approximative algorithms for configuration of multi-layer networks with protection. In: A. Jukan (ed.) DRCN 3rd Conference on Design of Reliable Communications Networks, pp. 228–235. Budapest (2001)

69. Gerstel, O., Lin, P., Sasaki, G.: Wavelength assignment in a WDM ring to minimize cost of embedded SONET rings. In: IEEE INFOCOM, pp. 94–101 (1998)

70. Gerstel, O., Ramaswani, R., Sasaki, G.: Cost-effective traffic grooming in WDM rings. IEEE/ACM Transactions on Networking **8**(5), 618–630 (2000)

71. Goldschmidt, O., Hochbaum, D., Levin, A., Olinick, E.: The SONET edge-partition problem. Networks **41**(1), 13–23 (2003)

72. Hong-Hsu, Y., Lee, S., Mukherjee, B.: Traffic grooming and delay constrained multicast routing in IP over WDM networks. In: IEEE International Conference on Communication (ICC), pp. 5246 – 5251. Beijing (2008). DOI 10.1109/ICC.2008.985

73. Hu, J.: Optimal traffic grooming for wavelength-division-multiplexing rings with all-to-all uniform traffic. OSA Journal of Optical Networks **1**(1), 32–42 (2002)

74. Huang, S., Dutta, R., Rouskas, G.: Traffic grooming in path, star, and tree networks: Complexity, bounds, and algorithms. IEEE Journal on Selected Areas in Communications **24**(4), 66–82 (2006)

75. Huiban, G., Pérennes, S., Syska, M.: Traffic grooming in WDM networks with multi-layer switches. In: IEEE International Conference on Communication (ICC), vol. 5, pp. 2896–2901 (2002). DOI 10.1109/ICC.2002.997370

76. Architecture for the automatically switched optical network (ASON) (2006). URL http://www.itu.int/rec/T-REC-G.8080/en

77. Jimenez, J., Gonzalez, O., Puype, B., Cinkler, T., Hegyi, P., Munoz, R., Martinez, R., Galan, F., Morro, R.: Multilayer traffic engineering experimental demonstrator in the Nobel-II project. In: BroadBand Europe. Antwerpen, Belgium (2007)

78. Kern, A., Moldován, I., Hegyi, P., Cinkler, T.: Ethernet over WDM: Optimization for resilience and scalability. In: DRCN 2007, 6th International Workshop on the Design of Reliable Communication Networks. La Rochelle, France (2007)

79. Kern, A., Somogyi, G., Cinkler, T.: On the gain of statistical multiplexing over traffic grooming. In: ICTON 2006, 8th International Conference on Transparent Optical Networks. Nottingham, United Kingdom (2006)

80. Kodialam, M. S., Lakshman, T. V.: Integrated dynamic IP and wavelength routing in IP over WDM networks. In: INFOCOM, pp. 358–366 (2001)

81. Köhn, M.: A new efficient online-optimization approach for SDH/SONET-WDM multi layer networks. In: OFC. Anaheim/CA, USA (2006)

82. Koutsoupias, E., Papadimitriou, C. H.: Worst-case equilibria. In: 16th Annual Symposium on Theoretical Aspects of Computer Science, pp. 404–413 (1999)

83. Ladányi, A., Cinkler, T., Hegyi, P., Szigeti, J.: SD-MLTE: State-dependent multi-layer traffic engineering. In: GMPLS WorkShop. Girona, Spain (2007)

84. Liu, L., Li, X., Wan, P. J., Frieder, O.: Wavelength assignment in a WDM rings to minimize SONET ADMs. In: INFOCOM'2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel-Aviv, Israel, pp. 1020–1025 (2000)

85. Maliosz, M., Cinkler, T.: Methods for optical VPN design over multifiber wavelength routing networks. In: IFIP ONDM 7th Conference on Optical Network Design and Modelling. Budapest, Hungary (2003)

86. Mannie (Ed.), E.: Generalized Multi-Protocol Label Switching (GMPLS) Architecture (2004). URL http://www.ietf.org/rfc/rfc3945.txt

87. Megyer, B., Cinkler, T., Szombat:, Z.: Setting up oVPNs with traffic grooming and protection. In: IFIP ONDM 2003, 7th Conference on Optical Network Design and Modelling. Budapest, Hungary (2003)

88. Megyer, B., Zsigmond, S., Szödényi, A., Cinkler, T.: Design of traffic grooming optical vpns obeying physical limitations. In: IEEE IFIP WOCN 2005. Dubai, UAE, March (2005)

89. Modiano, E., Lin, P.: Traffic grooming in WDM networks. IEEE Communications Magazine **39**(7), 124–129 (2001)

90. Muñoz, X., Sau, I.: Traffic Grooming in Unidirectional WDM Rings with Bounded Degree Request Graph. In: 34th International Workshop on Graph-Theoretic Concepts in Computer Science (WG) (2008)

91. Nash, J. F.: Equilibrium points in n-person games. In: Proceedings of the National Academy of Sciences, vol. 36, pp. 48–49 (1950)

92. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.): Algorithmic Game Theory. Cambridge University Press (2007)

93. Papadimitriou, D., Vigoureux, M., Shiomoto, K., Brungard, D., Roux, J. L. L.: Generalized multi-protocol label switching (GMPLS) protocol extensions for multi-layer and multi-region networks (MLN/MRN) (2008). URL http://www.ietf.org/internet-drafts/draft-ietf-ccamp-gmpls-mln-extensions-03.txt

94. Perényi, M., Breuer, J., Cinkler, T., Gáspár, C.: Grooming node placement in multilayer networks. In: 9th Conference on Optical Network Design and Modelling (ONDM). Milano, Italy (2005)

95. Perényi, M., Cinkler, T.: Joint grooming capability and wavelength number optimization. In: WTC 2006, World Telecommunications Congress. Budapest, Hungary (2006)

96. Perényi, M., Ladányi, A., Cinkler, T.: Joint grooming capability and wavelength number optimization. In: IEEE ICTON 2005, 7th International Conference on Transparent Optical Networks. Barcelona, Spain (2005)
97. Perényi, M., Soproni, P., Cinkler, T., Larrabeiti, D.: Regular reconfiguration of light-trees in multilayer optical networks. In: ONDM 2008, The 12th International Conference on Optical Networking Design and Modeling. Vilanova i la Geltru, Catalonia, Spain (2008)
98. Rajagopalan, B., Luciani, J., Awduche, D.: IP over optical networks: A framework, RFC 3717 (2004). URL http://www.ietf.org/rfc/rfc3717.txt
99. Rétvári, G., Fodor, P., Tapolcai, J., Cinkler, T.: Challenges of multi-layer traffic engineering in gmpls networks. In: 7th IEEE International Conference on Transparent Optical Networks (ICTON). Barcelona, Spain (2005)
100. Roughgarden, T.: Selfish Routing and the Price of Anarchy. MIT Press (2005)
101. Shalom, M., Unger, W., Zaks, S.: On the complexity of the traffic grooming problem in optical networks. In: 4th International Conference on Fun With Algorithms, Castiglioncello (LI), Tuscany, Italy (2007)
102. Shalom, M., Wong, P. W. H., Zaks, S.: Optimal on-line colorings for minimizing the number of ADMs in optical networks. In: Proceedings of the 21st Distributed Algorithms (DISC), pp. 435–449 (2007)
103. Shalom, M., Zaks, S.: A 10/7 + ε approximation scheme for minimizing the number of ADMs in SONET rings. In: First Annual International Conference on Broadband Networks, San-José, California, USA, pp. 254–262 (2004)
104. Shiomoto, K., Papadimitriou, D., Roux, J. L. L., Vigoureux, M., Brungard:, D.: Requirements for GMPLS-based multi-region and multi-layer networks (MRN/MLN) (2008). URL http://www.ietf.org/internet-drafts/draft-ietf-ccamp-gmpls-mln-reqs-11.txt
105. Solano, F., Caro, L., de Oliveira, J. C., Fabregat, R., Marzo, J. L.: G+: Enhanced traffic grooming in WDM mesh networks using lighttours. IEEE Journal on Selected Areas in Communications **25**(5), 1034–1047 (2007)
106. Somani, A.: Survivability and Traffic Grooming in WDM Optical Networks. Cambridge Press (2006)
107. Soproni, P., Perényi, M., Cinkler, T.: Grooming-enhanced multicast in multilayer networks. In: Optical Network Design and Modeling, *Lecture Notes in Computer Science*, vol. 4534, pp. 279–288. Springer, Athens, Greece (2007)
108. Su, D. H., Griffith., D. W.: Standards activities for MPLS over WDM networks. Optical Networks Magazine **1**(3) (2000)
109. Szigeti, J., Tapolcai, J., Cinkler, T., Henk, T., Sallai, G.: Stalled information based routing in multidomain multilayer networks. In: NETWORKS 2004, 11th International Telecommunication Network Planning Symposium. Vienna, Austria (2004)
110. Tapolcai, J., Cinkler, T.: Iterative multihop wavelength routing through decomposition. In: 9th International Telecommunication Network Planning Symposium (Networks). Toronto, Canada (2000)
111. Vigoureux, M., Berde, B., Andersson, L., Cinkler, T., Levrau, L., Colle, D., Fdez-Palacios, J., Jaeger, M.: Multilayer traffic engineering for GMPLS-enabled networks. Communications Magazine **43**(7), 44–50 (2005)
112. Wan, P. J., Calinescu, G., Liu, L., Frieder, O.: Grooming of arbitrary traffic in SONET/WDM BLSRs. IEEE Journal of Selected Areas in Communications **18**(10), 1995–2003 (2000)
113. Wang, J., Cho, W., Vemuri, V. R., Mukherjee, B.: Improved approaches for cost-effective traffic grooming in WDM ring networks: ILP formulations and single-hop and multihop connections. IEEE/OSA Journal of Lightwave Technology **19**(11), 1645–1653 (2001)
114. Ye, Y., Woesner, H., Chlamtac, I.: Waveband switching in light trail optical networks with dynamic traffic. OSA Journal of Optical Networking (2006). DOI 10.1364/JON.5.000701
115. Zhang, X., Qiao, C.: An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings. IEEE/ACM Transactions on Networking **8**(5), 608–617 (2000)

116. Zheng, J., Jue, J.: Waveband switching, routing, and grooming: introduction to the feature issue. OSA Journal of Optical Networking **6**(2), 217–218 (2007). DOI 10.1364/JON.6. 000217
117. Zhu, K., Mukherjee, B.: A review of traffic grooming in WDM optical networks: Architectures and challenges. Optical Networks Magazine **4**(2), 55–64 (2003)
118. Zhu, K., Zhu, H., Mukherjee, B.: Traffic Grooming in Optical WDM Mesh Networks. Springer (2005)
119. Zsigmond, S., Németh, G., Cinkler, T.: Mutual impact of physical impairments and grooming in multilayer networks. In: Optical Network Design and Modeling, *Lecture Notes in Computer Science*, vol. 4534, pp. 279–288. Springer, Athens, Greece (2007)
120. Zsigmond, S., Szödényi, A., Megyer, B., Cinkler, T., Tzanakaki, A., Tomkos, I.: A new method for considering physical impairments in multilayer routing. In: COST 291 - GBOU ONNA. Gent, Belgium (2006)

Chapter 3
Branch-and-Cut Techniques for Solving Realistic Two-Layer Network Design Problems

Sebastian Orlowski, Christian Raack, Arie M. C. A. Koster, Georg Baier, Thomas Engel, and Pietro Belotti

Abstract We study a planning problem arising in SDH/WDM multilayer telecommunication network design. The goal is to find a minimum cost installation of link and node hardware of both network layers such that traffic demands can be realized via grooming and a survivable routing. We present a mixed-integer programming formulation for a predefined set of admissible logical links that takes many practical side constraints into account, including node hardware, several bit rates, and survivability against single physical node or link failures. This model is solved using a branch-and-cut approach with problem-specific preprocessing, MIP-based heuristics, and cutting planes based on either of the two layers. On several realistic two-layer planning scenarios, we show that these ingredients can be very useful to reduce the optimality gaps in the multilayer context.

Key words: telecommunication networks, multilayer network design, preprocessing, integer linear programming, cutting planes, MIP-based heuristics

Sebastian Orlowski
atesio GmbH, Sophie-Taeuber-Arp-Weg 27, D-12205 Berlin, Germany, e-mail: `orlowski@atesio.de`

Christian Raack
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany, e-mail: `raack@zib.de`

Arie M. C. A. Koster
Lehrstuhl II für Mathematik, RWTH Aachen University, Wüllnerstr. zwischen 5 und 7, D-52062 Aachen, Germany, e-mail: `koster@math2.rwth-aachen.de`

Georg Baier
Siemens AG, Munich, Germany, e-mail: `georg.baier@siemens.com`

Thomas Engel
Nokia Siemens Networks GmbH & Co. KG, Munich, Germany, e-mail: `thomas.1.engel@nsn.com`

Pietro Belotti
Dept. of Industrial & Systems Engineering, Lehigh University, Bethlehem, PA, USA, e-mail: `belotti@lehigh.edu`

3.1 Introduction

Planning a telecommunication network is a nontrivial task. For a single network layer such as MPLS, SDH, or WDM, many mathematical models and algorithmic approaches have been proposed during the last 15 years. Links in an SDH network, for instance, may be equipped with a bandwidth of, say, 2.5 Gbit/s or 10 Gbit/s. This bandwidth is used to route several communication demands of lower granularity, like 155 Mbit/s. Dimensioning the links and routing the communication demands in the resulting network is a classical network design problem.

A practical telecommunication network, however, consists of several network layers which are embedded in each other. The bandwidth of an SDH link, for instance, can actually be realized by a capacitated lightpath in an underlying optical fiber network. The SDH and the WDM layer are highly interdependent: first, only a limited number of SDH links can traverse a given optical fiber, and second, the failure of a single optical fiber can disrupt many SDH links, and even more demand connections. In order to get a survivable network in practice, it is indispensible to plan both layers together.

More generally, the *two-layer network design problem* considered in this paper can be summarized as follows. Given is a set of network nodes together with potential connections between them. This network is called the *physical layer* and corresponds to the optical fiber network. On every fiber, a limited number of lightpath channels can be transmitted simultaneously, each of them corresponding to a capacitated path in the physical network. The nodes together with the lightpath connections form a so-called *logical network* on top of the physical one, as illustrated in Figure 3.1. In principle, any path in the physical network can be used for a lightpath, which leads to many parallel logical links. Even if the set of admissible lightpaths is often restricted to several short paths between each node-pair in practice, the resulting logical network is still much denser than a simple complete graph, which makes the network design problem hard to solve.
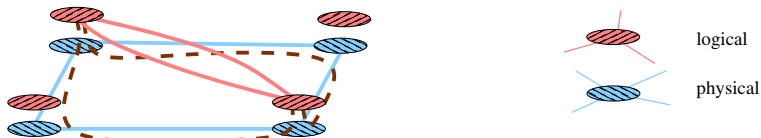
Fig. 3.1 Upper layer logical links (solid) correspond to paths (dashed) in the lower physical layer

Similar settings occur in many other technologies. An MPLS path, for example, can consist of links which are MPLS paths themselves. In an ATM/SDH setting, capacitated ATM links may be realized by an SDH radio link. Even if that radio link seems "less physical" than an optical fiber, it is called a physical link as well to indicate that it serves as part of a logical link. Similarly, the term "logical link" illustrates the fact that it looks like a direct link to its end nodes: first, it can be equipped with a discrete capacity, and second, traffic with lower granularity is sent

into the link at one end, extracted at the other end, and cannot be accessed inside. The actual physical representation of a logical link usually does not matter to its end nodes.

As this work was motivated by a project with Nokia-Siemens Networks (NSN) on SDH/WDM network planning, we will use that setting to explain our model and algorithm, which are actually in use at NSN in the strategic planning process now. But the concepts are more general and can, at least with slight modifications, be applied to many technological settings.

In our SDH/WDM scenario, a lightpath can be equipped with different bandwidths, and lower-rate traffic demands have to be routed via the lightpaths without exceeding their capacities. A demand may be 1+1-protected, i.e., twice the demand value must be routed such that in the case of any single physical link or node failure, at least the demand value survives. To terminate a lightpath, a sufficiently large electrical cross-connect (EXC) must be installed at both end nodes. The EXC converts the wavelength signal into an electrical SDH signal and extracts lower-rate traffic from it. The latter is either terminated at that node or recombined with other traffic to form new wavelength signals which are sent out on other lightpaths. This process is called *grooming*. The optimization goal is to minimize total installation cost.

Like in any other publication where an integrated two-layer model is actually used for computations, we do not explicitly assign wavelengths to the lightpaths because finding a suitable wavelength assignment is an extremely hard problem on its own. Instead, we make sure that the maximum number of lightpaths on each fiber is not exceeded, and propose to solve the wavelength assignment and converter installation problem in a subsequent step, as successfully done in [19]. It has been shown in [20] that such an approach causes at most a marginal increase in the overall installation cost in practical instances.

Already, the optimal design of a single layer network is a challenging task that has been considered by many research groups; see for instance [4, 14, 28] and references therein. A branch-and-cut algorithm enhanced by user-defined, problem-specific cutting planes has been proved to be a very successful solution approach in this context. The combined optimization of two layers significantly increases the complexity of the planning task. In most previous publications, mixed-integer programming techniques have been used for designing a logical layer with respect to a fixed physical layer [5, 11, 12] or for solving an integrated two-layer planning problem with some simplifying assumptions, like no node hardware or wavelength granularity demands [15, 21]. Knippel and Lardeux [17] and Fortz and Poss [13] have modeled the two-layer network design problem using metric inequalities for both network layers. Recently, Belotti et al. [6] have used a Lagrangean approach for a two-layer network design problem with simultaneous mean demand values and nonsimultaneous peak demand values. Raghavan and Stanojevic [29] consider the case where all logical links are eligible and develop a branch-and-price algorithm with respect to a fixed physical layer for the case of unprotected demands and one facility on the logical links. Orlowski et al. [25] present several heuristics for a two-layer network design problem, which solve a restricted version of the original problem as a sub-MIP within a branch-and-cut framework. In a recent paper [18],

we have presented a mathematical model for the described planning problem with a predefined set of logical links. It includes node hardware, several bit rates on the logical links, and survivability against physical node and link failures. To our knowledge, this was the first time that so many practically relevant side constraints, and in particular, multilayer survivability, were taken into account in an integrated two-layer planning model.

We have solved this model using a branch-and-cut approach with problem-specific preprocessing, user-defined cutting planes, and heuristics. This chapter combines results from [25] and [18]. In addition to the preprocessing and cutting planes presented in [18], we have also adapted the MIP-based primal heuristics from [25] to the full planning problem and call them at various places during the branch-and-cut tree. The algorithm is tested on several network instances provided by Nokia Siemens Networks. The paper is structured as follows. In Section 3.2, we present and discuss our mixed-integer programming model. Section 3.3 describes our MIP-based primal heuristics used within the branch-and-cut algorithm. In Section 3.4, we describe the cutting planes used and state some known results about their strength. Computational results are provided in Section 3.5. We conclude with Section 3.6.

We assume that that reader has basic knowledge of mixed-integer programming and branch-and-cut techniques; good introductions to this topic are [24] and [31].

3.2 Mathematical Model

3.2.1 Mixed-Integer Programming Model

We will now introduce the mixed-integer programming (MIP) model on which our cutting planes are based. Afterwards, we will describe some basic preprocessing steps that we have applied to strengthen the formulation.

Parameters

The physical network is represented by an undirected graph (V, E). The logical network is modeled by an undirected graph (V, L) with the same set of nodes and a fixed set L of admissible logical links. Each logical link represents an undirected path in the physical network. In consequence, any two nodes $i, j \in V$ may be connected by many parallel logical links corresponding to different physical paths, collected in the set $L_{ij} = L_{ji}$. Looped logical links are forbidden, i.e., $L_{ii} = \emptyset$ for all $i \in V$. Let $\delta_L(i) = \cup_{j \in V} L_{ij}$ be the set of all logical links starting or ending at i. Eventually, $L_e \subseteq L$ denotes the set of logical links containing edge $e \in E$, and likewise, $L_i \subseteq L$ refers to the set of logical links containing node $i \in V$ as an inner node.

We consider different types of capacities for logical links, physical links, and nodes. Each logical link $\ell \in L$ has a set M_ℓ of available capacity modules, each of them with a cost of $\kappa_\ell^m \in \mathbb{R}_+$ and a base capacity (bit rate) of $C_\ell^m \in \mathbb{Z}_+$ that can be installed on ℓ in integer multiples. Similarly, every node $i \in V$ has a set M_i of node modules (representing different EXC types), at most one of which may be installed at i. Module $m \in M_i$ provides a switching capacity of $C_i^m \in \mathbb{Z}_+$ (e.g., in bits per second) at a cost of $\kappa_i^m \in \mathbb{R}_+$. On a physical link $e \in E$, a fiber may be installed at a cost of $\kappa_e \in \mathbb{R}_+$. Each fiber supports up to $B \in \mathbb{Z}_+$ lightpaths.

For the routing part, a set H of undirected point-to-point communication demands is given, which may be *protected* or *unprotected*. Protected demands are expected to survive any single physical node or link failure, whereas unprotected demands are allowed to fail. Each demand $h \in H$ has a source node, a target node, and a demand value d_h to be routed between these two nodes. Without loss of generality, we may assume the demands to be directed in an arbitrary way. For 1+1-protected demands, d_h refers to twice the original demand value that would have to be routed if the demand were unprotected. Adding constraints that limit the amount of flow for a protected commodity through a node or physical link to $\frac{1}{2}d_h$ guarantees that at least the original demand survives any single physical link or node failure. This survivability model, called *diversification* [3], is a slight relaxation of 1+1-protection, but its solutions can often be transformed into 1+1-solutions.

From the demands, two sets K^p and K^u of protected and unprotected commodities are constructed, where $K = K^p \cup K^u$ denotes the set of all commodities. With every commodity $k \in K$ and every node $i \in V$, a net demand value $d_i^k \in \mathbb{Z}$ is associated such that $\sum_{i \in V} d_i^k = 0$. Every *protected commodity* $k \in K^p$ consists of a single 1+1-protected point-to-point demand, i.e., $d_i^k \neq 0$ only for the source and target node of the demand. In contrast, *unprotected commodities* $k \in K^u$ are derived by aggregating unprotected point-to-point demands at a common source node. Summarizing, every commodity $k \in K$ has a unique source node $s^k \in V$. Unprotected commodities may have several target nodes, whereas protected commodities have a unique target $t^k \in V$. The (undirected) emanating demand of a node $i \in V$, i.e., the total demand value starting or ending at node i, is given by $d_i = \sum_{k \in K} |d_i^k|$. The demand value d^k of a commodity is defined as the demand for k emanating from its source node, i.e., $d^k = d_{s^k}^k > 0$. Notice that for protected commodities, this value is twice the requested bandwidth to ensure survivability.

Variables

The model comprises four classes of variables representing the flow and different capacity types. First, for a logical link $\ell \in L$ and a module $m \in M_\ell$, the logical link capacity variable $y_\ell^m \in \mathbb{Z}_+$ represents the number of modules of type m installed on ℓ. For a physical link $e \in E$, the binary physical link capacity variable $z_e \in \{0, 1\}$ indicates whether e is equipped with a fiber or not. Similarly, for a node $i \in V$ and a node module $m \in M_i$, the binary variable $x_i^m \in \{0, 1\}$ denotes whether module m is installed at node i or not. Eventually, the routing of the commodities is modeled

by flow variables. In order to model diversification of protected commodities, we need fractional flow variables $f_{\ell,ij}^k, f_{\ell,ji}^k \in \mathbb{R}_+$ representing the flow for commodity $k \in K$ on logical link $\ell \in L_{ij}$ directed from i to j and from j to i, respectively. For notational convenience, $f_\ell^k = f_{\ell,ij}^k + f_{\ell,ji}^k$ denotes the total flow for $k \in K$ on $\ell \in L_{ij}$ in both directions.

In our model, a flow variable $f_{\ell,ij}^k$ for commodity k and logical link $\ell \in L_{ij}$ is omitted if any of the following conditions is satisfied: (i) $j = s^k$, (ii) $k \in K^p$ and $i = t^k$, and (iii) $k \in K^p$ and ℓ contains the source or target node of k as an inner node. The first two types of variables represent flow into the unique source node or out of the unique target node of a protected commodity. They are not generated in order to reduce cycle flows in the edge-flow formulation. For aggregated unprotected commodities, we have to allow flow from one target node to another, and thus flow out of target nodes. The third type of variable would allow flow to be routed through an end node u of a protected commodity without terminating at that node, and then back to u on another logical link. As such routings are not desired in practice, we exclude flow variables whose logical link contains an end node of the corresponding commodity as an inner node. Again, in the unprotected case, such variables have to be admitted because commodities may consist of several aggregated demands.

Objective and Constraints

The objective and constraints of our MIP model read as follows:

$$\min \quad \sum_{i \in V} \sum_{m \in M_i} \kappa_i^m x_i^m + \sum_{\ell \in L} \sum_{m \in M_\ell} \kappa_\ell^m y_\ell^m + \sum_{e \in E} \kappa_e z_e \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{j \in V} \sum_{\ell \in L_{ij}} (f_{\ell,ij}^k - f_{\ell,ji}^k) \qquad\qquad = d_i^k \qquad \forall i \in V, \forall k \in K \tag{3.1b}$$

$$\sum_{m \in M_\ell} C_\ell^m y_\ell^m - \sum_{k \in K} f_\ell^k \qquad\qquad \geq 0 \qquad \forall \ell \in L \tag{3.1c}$$

$$\sum_{\ell \in L_i} f_\ell^k + \sum_{\ell \in \delta_L(i)} \frac{1}{2} f_\ell^k \qquad\qquad \leq \frac{1}{2} d^k \qquad \forall i \in V, \forall k \in K^p \tag{3.1d}$$

$$f_{\ell,s^k,t^k}^k \qquad\qquad\qquad \leq \frac{1}{2} d^k \qquad \begin{matrix} \forall k \in K^p, \\ \ell = e = \{s^k, t^k\} \end{matrix} \tag{3.1e}$$

$$\sum_{m \in M_i} x_i^m \qquad\qquad\qquad\qquad \leq 1 \qquad \forall i \in V \tag{3.1f}$$

$$2 \sum_{m \in M_i} C_i^m x_i^m - \sum_{\ell \in \delta_L(i)} \sum_{m \in M_\ell} C_\ell^m y_\ell^m \qquad \geq d_i \qquad \forall i \in V \tag{3.1g}$$

$$B z_e - \sum_{\ell \in L_e} \sum_{m \in M_\ell} y_\ell^m \qquad\qquad \geq 0 \qquad \forall e \in E \tag{3.1h}$$

$$f_{\ell,ij}^k, f_{\ell,ji}^k \in \mathbb{R}_+, \ y_\ell^m \in \mathbb{Z}_+, \ x_i^m, z_e \quad \in \{0,1\} \tag{3.1i}$$

The objective (3.1a) aims at minimizing the total installation cost. The flow conservation (3.1b) and capacity constraints (3.1c) describe a multi-commodity flow and modular capacity assignment problem on the logical layer. For protected commodities, the flow diversification constraints (3.1d) restrict the flow through an intermediate node to half the demand value. In this way, the original demand is guaranteed to survive single node failures as well as single physical link failures, except for the direct physical link between source s^k and target t^k. This exception is covered by the variable bound (3.1e). In fact, to reduce cycle flows in the LP, we set an upper bound of d^k and $\frac{1}{2}d^k$ on *all* flow variables for unprotected and protected commodities, respectively. The generalized upper bound constraints (3.1f) guarantee that at most one node module is installed at each node. The node switching capacity constraints (3.1g) ensure that the switching capacity of the network element installed at a node is sufficient for all traffic that can potentially be switched at that node. Since all traffic is counted twice, it is compared to twice the installed node capacity. Finally, the physical link capacity constraints (3.1h) make sure that the maximum number of modules on a physical link is not exceeded, and set the physical link capacity variables to 1 whenever a physical link is used.

Discussion of the Model

Several design choices in our model deserve a brief discussion. First, we assume a fractional multi-commodity flow on the logical layer although SDH requires an integer routing in practice. This is motivated by our observations that in good solutions, the routing is often nearly integer even if this is not required, and that relaxing the integrality conditions on the flow variables significantly reduces the computation times. If an integral routing is indispensable, it can be obtained in a postprocessing step, which usually does not deteriorate the cost of the solutions very much if properly done. Notice that the lower bound computed for the model with fractional flow can also be used to assess the quality of the postprocessed integral solutions.

Second, we assume a predefined set of logical links for computational reasons. Considering all possible physical paths as logical links in combination with the practical side constraints and the survivability requirements would ask for a branch-and-cut-and-price approach with a nontrivial pricing problem already in the root node. Such an approach can only be successful if the problem with a limited set of logical links can be solved efficiently. For a branch-and-price approach that deals with all possible logical links on a fixed physical layer using a simplified model without survivability, the reader is referred to [29].

3.2.2 Preprocessing

To strengthen the formulation, we now describe the preprocessing steps applied to the model presented in Section 3.2. In addition to these steps, we have used probing

techniques to set further bounds on variables [30]. Probing is nowadays part of any modern MIP solver like SCIP [1] or CPLEX [16].

- If a physical link $e \in E$ has zero fiber cost, the variable z_e can be fixed to 1.
- Obviously, at least the emanating demand must be switched at every node. Consequently, an EXC must be installed at every demand end node $i \in V$, so the node GUB inequality (3.1f) can be changed to an equality at such nodes: $\sum_{m \in M_i} x_i^m = 1$. This strengthens the LP relaxation significantly.
- For the same reason, node modules whose switching capacity is smaller than the emanating demand at a node cannot be installed at that node. Consequently, if $C_i^m < d_i$ for some node $i \in V$ and a node module $m \in M_i$, the corresponding variable x_i^m can be removed from the MIP formulation. This often leads to more integral x_i^m variables in the LP relaxation and to better LP values, especially when combined with the previous rule.
- Two bounds on logical link module variables can be derived from the fiber capacity bounds and the total demand in the network. Both bounds are usually not tight in the LP relaxation, but may help the MIP solver in deriving further relations between the variables to strengthen other bounds.

  First, as no more than B channels can be routed through a given physical link $e \in E$, every logical link module variable can be bounded by B. Second, the amount of flow that can be routed through any logical link $\ell \in L$ is bounded by the total unprotected demand plus half the protected demand in the network (except for undesired cycle flow in the edge-flow formulation). Consequently, the number of modules of type $m \in M_\ell$ that possibly needs to be installed on ℓ is bounded by

$$y_\ell^m \leq \left\lceil \frac{1}{C_\ell^m} \left(\sum_{h \in H^u} d_h + \sum_{h \in H^p} \frac{1}{2} d_h \right) \right\rceil .$$

- Sometimes it is evident that in any optimal solution, a small link module will not be installed more than a given number of times on a link because a larger module provides the same or more capacity at a lower price. More precisely, consider a link $\ell \in L$ and two of its capacity modules $m_1, m_2 \in M_\ell$ such that $C_\ell^{m_1} \leq C_\ell^{m_2}$. If the relation

$$r = \frac{\kappa_\ell^{m_2}}{\kappa_\ell^{m_1}} \leq \frac{C_\ell^{m_2}}{C_\ell^{m_1}}$$

holds, then at most r modules of type m_1 will be installed in any optimal solution because r modules of type m_1 incur the same cost as one unit of type m_2, but the latter provides the same or more capacity. Furthermore, even if equality holds in the above relation, one large module is preferable to several smaller ones because every module uses one physical channel, independently of its bit rate. Consequently, the variable bound $y_\ell^{m_1} \leq \lceil r - 1 \rceil$ can be added to the formulation. It cuts off some non-optimal solutions and maybe some optimal ones

(if equality holds), but always leaves at least one optimal solution if one exists. Notice that the value $\lceil r-1 \rceil$ is exactly $r-1$ if r is integer, and $\lfloor r \rfloor$ otherwise.

3.3 MIP-Based Heuristics Within Branch-and-Cut

We solve the mixed-integer programming formulation using the branch-and-cut framework SCIP 0.90 [2]. In addition, we have implemented several heuristics to construct feasible network configurations based on integer or fractional solutions. At every node of the search tree, SCIP generates cutting planes and calls both our heuristics and some of its own to identify feasible integer solutions. If a new best solution is identified, it is added to SCIP's solution pool such that it can be used by other heuristics which take feasible solutions as a basis for their work. We will now describe our heuristics and their use within the branch-and-cut framework.

Our MIP-based heuristics address two major subtasks. GROOMCAPMIP and GROOMCAPHEUR solve the grooming and capacity installation subproblem for a given routing exactly and heuristically, respectively, whereas REROUTINGMIP computes a routing within certain link capacities, trying to reduce the required capacity at the same time. By construction, the MIP-based heuristics can easily be adapted to include additional planning requirements, such as node hardware or survivability constraints.

3.3.1 Computing Capacities over a Given Flow

GROOMCAPMIP

The GROOMCAPMIP procedure addresses the grooming and capacity assignment subproblem for a given routing by solving a MIP. For a logical link $\ell \in L_{ij}$, let $f_\ell^* = \sum_{k \in K} \sum_{\ell \in L} (f_{\ell,ij}^k + f_{\ell,ji}^k)$ be the total flow on ℓ in an integer or LP solution (after removing possible cycle flows). We construct a sub-MIP of the original formulation (3.1a)–(3.1i) that contains logical and physical capacity variables but no routing information:

$$\min \left\{ (3.1a) \text{ subject to } (3.1f) - (3.1h), \ \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq \lceil f_\ell^* \rceil \ \forall \ell \in L, \ z_e, y_\ell^m \in \mathbb{Z}_+ \right\}.$$

Using SCIP's branch-and-cut algorithm, this sub-MIP is solved as an improvement heuristic every time a new best solution is identified, trying to reduce link capacity cost based on the given routing. As the focus of the sub-MIP is on feasible solutions and not on the lower bound, we disable cut generation and expensive heuristics in the subproblem and impose a node limit of 20,000 and a stall node limit of 10,000, i.e., the sub-MIP is stopped if either a total of 20,000 branch-and-cut nodes has

been computed or if the primal bound could not be improved during the last 10,000 nodes.

GROOMCAPHEUR

In contrast to the GROOMCAPMIP algorithm, which solves the grooming and capacity assignment problem exactly, the fast and simple GROOMCAPHEUR procedure addresses this problem heuristically by decomposition. Again, let f_ℓ^* be the total flow on logical link $\ell \in L$ in an integer or LP solution after removing cycle flows. Installing capacities on ℓ at minimum cost with a lower bound of f_ℓ^* can be formulated as an integer knapsack problem:

$$\min \left\{ \sum_{m \in M_\ell} \kappa_\ell^m y_\ell^m \text{ subject to } \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq \lceil f_\ell^* \rceil, \ y_\ell^m \in \mathbb{Z}_+ \right\}.$$

For $|M_\ell| = 1$ this knapsack problem is trivial to solve. Otherwise, it is solved heuristically for each logical link $\ell \in L$ using a greedy algorithm, taking the maximum capacity of each physical link into account. In a second step, node capacities are installed as much as needed for the given link capacities (if possible). As this heuristic runs very fast, we call it at every branch-and-cut node to construct feasible solutions from the current LP solution.

3.3.2 Rerouting Flow to Reduce Capacities

REROUTINGMIP

The REROUTINGMIP heuristic determines a routing together with a minimum-cost capacity installation subject to an upper capacity bound on the logical links. More precisely, given an upper bound U_ℓ^* on the capacity of each logical link $\ell \in L$, REROUTINGMIP solves the following problem using SCIP's branch-and-cut capabilities:

$$\min \left\{ (3.1a) \text{ subject to } (3.1b)–(3.1i), \ \sum_{m \in M_\ell} C_\ell^m y_\ell^m \leq U_\ell^* \ \forall \ell \in L \right\}.$$

With small U_ℓ^*, this problem is much easier to solve than the original problem. By setting U_ℓ^* to the total capacity of link $\ell \in L$ in an integer solution, REROUTINGMIP can be used as an improvement algorithm that tries to reduce capacities by rerouting flow. This generalizes the rerouting step in the iterative heuristics proposed in [15, 21], making it independent of the ordering of the demands.

We employ REROUTINGMIP not as an improvement heuristic but as a construction algorithm. Given some value $\kappa \geq 1$ and an LP solution with total logical link

capacities $y_\ell^* = \sum_{m \in M_\ell} C_\ell^m y_\ell^{m*}$, we solve the above sub-MIP with $U_\ell^* := C^0 \left\lceil \frac{\kappa}{C^0} y_\ell^* \right\rceil$ where C^0 is the smallest module capacity installable on ℓ. If the installable capacities form a divisibility chain (which is often the case in practical applications), U_ℓ^* is the smallest installable integer capacity greater than or equal to κy_ℓ^*. Obviously, a higher value of κ augments the solution space of the subproblem, allowing for better solutions but also making it harder to solve. Experimenting with different values, we found that $\kappa = 2$ often allowed us to quickly determine good solutions in the sub-MIP.

As the REROUTINGMIP algorithm consumes much more time than the other heuristics, we restrict its application to the LP solution at the end of the branch-and-cut root node. In the sub-MIP (as well as in the original problem), good solutions are often found within the first few branch-and-bound nodes, whereas much time is spent afterwards on proving optimality of the solution. Hence, we disable cut generation and expensive heuristics in the subproblem, and we impose a node limit of 10,000 nodes and a stall node limit of 5,000 nodes. To increase the chance of finding good solutions, we also apply the GROOMCAPHEUR and GROOMCAPMIP algorithms within the sub-MIP, which tends to improve the overall solution quality.

3.4 Cutting Planes

Backed by theoretical results of polyhedral combinatorics, cutting plane procedures have proved to be a feasible approach to improve the performance of mixed-integer programming solvers for many single-layer network design problems. In this section we show how an appropriate selection of these inequalities can be adapted to our problem setting. Their separation within a branch-and-cut algorithm, i.e., the problem to find a violated inequality that cuts off a fractional LP solution or to determine that no such inequality exists, is only briefly summarized here; details can be found in [18].

3.4.1 Cutting Planes on the Logical Layer

On the logical layer, we consider *cutset inequalities* and *flow-cutset inequalities*. These cutting planes have, for instance, been studied in [4, 8, 10, 22, 28] for a variety of network settings (e.g., directed, undirected, and bidirected link models, single or multiple capacity modules) and have been successfully used within branch-and-cut algorithms for capacitated single-layer network design problems [7, 8, 14, 28].

To be precise, the inequalities on the logical layer are valid for the polyhedron P defined by the multi-commodity flow constraints (3.1b) and the capacity constraints (3.1c). That is,

$$P = \text{conv} \left\{ (f, y) \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{n_2} \mid (f, y) \text{ satisfies } (3.1b), (3.1c) \right\},$$

where $n_1 = 2|K||L|$ and $n_2 = \sum_{\ell \in L} |M_\ell|$. As P is a relaxation of the model discussed in Section 3.2, the inequalities are also valid for that model.

We introduce the following notation. For any subset $\emptyset \neq S \subset V$ of nodes, let

$$L_S = \left\{ \ell \in L \mid \ell \in L_{ij}, \ i \in S, \ j \in V \setminus S \right\}$$

be the set of logical links having exactly one end node in S. Furthermore, define $d_S^k = \sum_{i \in S} d_i^k \geq 0$ to be the total demand value to be routed over the cut L_S for commodity $k \in K$. By reversing the direction of demands and exchanging the corresponding flow variables, we may w. l. o. g. assume that $d_S^k \geq 0$ for all $k \in K$ (i.e., the commodity is directed from S to $V \setminus S$, or the end nodes of k are either all in S or all in $V \setminus S$). This reduction is done implicitly in our code. More generally, let $d_S^Q = \sum_{k \in Q} d_S^k$ denote the total demand value to be routed over the cut L_S for all commodities $k \in Q$.

Mixed-Integer Rounding (MIR)

In order to derive strong valid inequalities on the logical layer we aggregate model inequalities and apply a strengthening of the resulting base inequalities; this is known as *mixed-integer rounding* (MIR). It exploits the integrality of the capacity variables. Further details on mixed-integer rounding can be found in [23], for instance.

Let $a, c, d \in \mathbb{R}$ with $c > 0$ and $\frac{d}{c} \notin \mathbb{Z}$, and $a^+ = \max(0, a)$. Furthermore, let

$$r_{a,c} = a - c(\lceil \tfrac{a}{c} \rceil - 1) > 0$$

be the remainder of the division of a by c if $\frac{a}{c} \notin \mathbb{Z}$, and c otherwise. Now assume that $d/c \notin \mathbb{Z}$ and consider the subadditive MIR functions

$$F_{d,c} : \mathbb{R} \to \mathbb{R} : a \mapsto \left\lceil \tfrac{a}{c} \right\rceil r_{d,c} - (r_{d,c} - r_{a,c})^+$$

and $\bar{F}_{d,c}(a) = \lim_{t \searrow 0} \frac{F_{d,c}(at)}{t} = a^+$. Given any valid inequality for our problem, applying $F_{d,c}$ and $\bar{F}_{d,c}$ to the integer and continuous variables, respectively, yields another valid inequality [24]. Moreover, the resulting coefficients are integral (if a, c, and d are integral) and $|F_{d,c}(a)|, |\bar{F}_{d,c}(a)| \leq |a|$, as shown in [28]. Both features are desirable from a numerical point of view. For more details and explanations see [28].

Cutset Inequalities

Let L_S be a cut in the logical network as defined above. Obviously, the total capacity on the cut links L_S must be sufficient to accommodate the total demand over the cut:

$$\sum_{\ell \in L_S} \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq d_S^K. \tag{3.2}$$

Since all coefficients are nonnegative in inequality (3.2) and $y_\ell^m \in \mathbb{Z}_+$, we can round down all coefficients to the value of the right-hand side (if larger). For notational convenience we assume from now on $C_\ell^m \leq d_S^K$ for all $\ell \in L_S$ and $m \in M_\ell$. Mixed-integer rounding exploits the integrality of the capacity variables. Setting $c > 0$ to any of the available capacities on the cut and applying the MIR function $F_c = F_{d_S^K, c}$ to the coefficients and the right-hand side of inequality (3.2) results in the *cutset inequality*

$$\sum_{\ell \in L_S} \sum_{m \in M_\ell} F_c(C_\ell^m) y_\ell^m \geq F_c(d_S^K). \tag{3.3}$$

A crucial necessary condition for inequality (3.3) to define a facet for P is that the two subgraphs defined by the network cut be connected, which is trivially fulfilled if L contains logical links between all node pairs.

Given a fractional LP solution, we look for violated MIR inequalities by setting weights on the logical links based on the primal and dual LP solutions, shrinking the logical graph with respect to these weights until only a small number of nodes (say, four or five) remain. In this shrunken graph, we enumerate all cuts, construct the corresponding MIR inequality, and test it for violation. For details, the interested reader is referred to [18].

Flow-Cutset Inequalities

Cutset inequalities can be generalized to flow-cutset inequalities, which have non-zero coefficients also for flow variables. Like cutset inequalities, flow-cutset inequalities are derived by aggregating capacity and flow conservation constraints on a logical cut L_S and applying a mixed-integer rounding function to the coefficients of the resulting inequality. However, the way of aggregating the inequalities is more general. Various special cases of flow-cutset inequalities have been discussed in [4, 8, 10, 28]. Necessary and sufficient conditions for flow-cutset inequalities to define a facet of P can be found in [28].

Consider fixed nonempty subsets $S \subset V$ of nodes and $Q \subseteq K$ of commodities. Assume that logical link $\ell \in L_S$ has end nodes $i \in S$ and $j \in V \setminus S$. We will denote by $f_{\ell,-}^k = f_{\ell,ji}^k$ inflow into S on ℓ while $f_{\ell,+}^k = f_{\ell,ij}^k$ refers to outflow from S on ℓ. We now construct a base inequality to which a suitable mixed-integer rounding function will be applied. First, we obtain a valid inequality from the sum of the flow conservation constraints (3.1b) for all $i \in S$ and all commodities $k \in Q$:

$$\sum_{\ell \in L_S} \sum_{k \in Q} (f_{\ell,+}^k - f_{\ell,-}^k) \geq d_S^Q$$

Given a subset $L_1 \subseteq L_S$ of cut links and its complement $\bar{L}_1 = L_S \setminus L_1$ with respect to the cut, we can relax the above inequality by omitting the inflow variables and by replacing the flow by the capacity on all links in L_1:

$$\sum_{\ell \in L_1} \sum_{m \in M_\ell} C_\ell^m y_\ell^m + \sum_{\ell \in \bar{L}_1} \sum_{k \in Q} f_{\ell,+}^k \geq d_S^Q. \tag{3.4}$$

Again, we may assume $C_\ell^m \leq d_S^K$ for all $\ell \in L_1$ and $m \in M_\ell$. Let $c > 0$ be the capacity of a module available on the cut and define $F_c = F_{d_S^Q,c}$ and $\bar{F}_c = \bar{F}_{d_S^Q,c}$. Applying these functions to the base inequality (3.4) results in the *flow-cutset inequality*

$$\sum_{\ell \in L_1} \sum_{m \in M_\ell} F_c(C_\ell^m) y_\ell^m + \sum_{\ell \in \bar{L}_1} \sum_{k \in Q} f_{\ell,+}^k \geq F_c(d_S^Q). \tag{3.5}$$

Notice that $\bar{F}_c(1) = 1$, so the coefficients of the flow variables remain unchanged. This inequality can be generalized to a flow-cutset inequality also containing in-flow variables [28]. By choosing $L_1 = L_S$ and $Q = K$, inequality (3.5) reduces to inequality (3.3).

For separating a flow-cutset inequality, a suitable set S of nodes, a subset Q of commodities, a capacity c, and a partition $(L_1, \bar{L}_1)$ of the cut links L_S have to be chosen. We apply two different separation heuristics. The first heuristic considers commodity subsets Q with a single commodity $k \in K$ and node sets S consisting of one or two end nodes of k. After fixing S and k and choosing an available capacity $c > 0$ on the cut, a partition of the cut links that maximizes the violation for flow-cutset inequalities can be obtained in linear time; see [4, 18]. The second, more time-consuming heuristic finds a most violated flow-cutset inequality for a fixed single commodity $k \in K$ and a fixed capacity c using a Min-Cut Algorithm; see [4].

3.4.2 Cutting Planes on the Physical Layer

If the fixed-charge cost values κ_e are zero, then the corresponding variables z_e can be assumed equal to 1 in any optimal solution. If, on the other hand, this cost is positive, the variables will take on fractional values in linear programming (LP) relaxations. By the demand routing requirements, we know that certain pairs of nodes have to be connected not only on the logical layer but also on the physical layer. Consequently, the variables z_e have to satisfy certain connectivity constraints. Note that information of the physical layer is combined with the demands here, skipping the intermediate logical layer.

Connectivity problems have been studied on several occasions, in particular in the context of the Steiner Tree problem and fixed-charge network design, e.g., [9, 27]. Let $S \subset V$ be a set of nodes and $\delta(S)$ be the corresponding cut in the physical network. If some demand has to cross the cut, then the inequality

$$\sum_{e \in \delta(S)} z_e \geq 1 \tag{3.6}$$

ensures that at least one physical link is installed on the cut. If a protected demand has to cross the cut, the right-hand side can even be set to 2 because the demand must be routed on at least two physically disjoint paths.

If the demand graph (defined by the network nodes and edges corresponding to traffic demands) has p connected components (usually $p = 1$), then

$$\sum_{e \in E} z_e \geq |V| - p \qquad\qquad (3.7)$$

is valid, because the installed physical links can consist of at most p connected components as well, each one being at least a tree. If protected demands exist and the demand graph is connected, inequality (3.7) can be strengthened by setting the right-hand side to $|V|$. If protected demands exist for all demand end nodes, this inequality is dominated by the inequalities (3.6) for all demand end nodes as single node subsets.

As the number of inequalities (3.6) and (3.7) is very small, we do not separate them but just add them all in the beginning of the branch-and-bound process.

3.5 Computational Results

3.5.1 Test Instances and Settings

For our computational experiments we used the network instances summarized in Table 3.1. In addition to the number of nodes and physical and logical links, the number $|H|$ of communication demands is given, from which the commodities were constructed ($|K| = |V| - 1$ if all demands are unprotected and $|K| = |H|$ if all demands are protected). Further, we report the number $|M_i|$ of node modules installable at each node and the size of the installable logical link modules. Finally, Table 3.1 indicates whether the instance has physical link cost or not. The first three instances are realistic scenarios provided by Nokia Siemens Networks, whereas the small ring network Ring7 has been constructed out of the larger instance Ring15 in order to study the effect of the cutting planes on the number of branch-and-cut nodes, needed to prove optimality.

Table 3.1 Network instances used for testing cutting planes

instance	$|V|$	$|E|$	$|L|$	$|H|$	$|M_i|$	$C_\ell^1, C_\ell^2, C_\ell^3$	physical cost?
Germany17	17	26	674	121	16	1, 4, 16	no
Germany17-fc	17	26	564	121	16	1, 4, 16	yes
Ring15	15	16	184	78	5	16, 64, 256	no
Ring7	7	8	32	10	5	16, 64, 256	no

Germany17 and Germany17-fc are based on a physical 17-node German network available at SNDlib [26]. In both networks, the set of admissible logical links consists of three to five short paths in the physical network between each pair of nodes. Ring15 consists of a physical ring with a chord representing a regional subnetwork connected to a larger national network. The set of logical links consists basically of the two possible logical links for each node pair, one in each physical direction of the ring. Ring7 has been constructed from Ring15 by successively removing nodes with the smallest emanating demand value. Because in our ring instances every node is a demand end node and the demand graph is connected, nearly all physical links have to be used in any feasible solution. We thus do not consider ring variants with physical link cost because doing so would only add a constant to the objective function. In all networks, up to three capacity modules corresponding to 2.5, 10, and 40 Gbit/s can be installed on each logical link depending on its physical path length.

All computations were done on a Linux-operated machine with a 2×3 GHz Intel P4 processor and 2 GB of memory. In a first series of test runs, we assumed unprotected demands with physical fibers supporting $B = 40$ wavelengths. In a second series, we made all demands 1+1-protected, assuming $B = 80$ wavelengths in order to allow for feasible solutions with the doubled demand values.

The focus of our computational results is on the effect of the cutting planes, which is discussed in Section 3.5.2 for unprotected networks and in Section 3.5.3 for networks with 1+1 protection. In the corresponding tests, we have always used the preprocessing steps described in Section 3.2.2 and the primal heuristics from Section 3.3 unless otherwise stated. The effect of the heuristics and our preprocessing is discussed in Section 3.5.4.

3.5.2 Unprotected Demands

As cutting planes are primarily thought to increase the lower bound of the LP relaxation, we first consider the effect of the different types of cutting planes on the lower bound at the branch-and-bound root node. We separated the classes cutset inequalities, flow-cutset inequalities, and fixed-charge inequalities on their own as well as all together. Figure 3.2 shows the improvement over time of the lower bound in the root node of the search tree for all test instances. The solid red line at the top marks the value of the best-known solution, which cannot be exceeded by the dual bound curves. The line "no cutting planes" refers to the dual bound with SCIP's built-in general-purpose cuts only.

It can be seen that in the two Germany17 instances and on the small ring network, our cutting planes reduce the gap between the lower bound and the best-known solution at the root node by 50%–75%. In all three problem instances, flow-cutset inequalities performed better than cutset inequalities, which is in contrast to the results presented by Raack et al. [28] for a single-layer problem. There might be several reasons for this effect. A good candidate is the structural difference between single-layer networks and the logical layer in multilayer problems: the logical layer
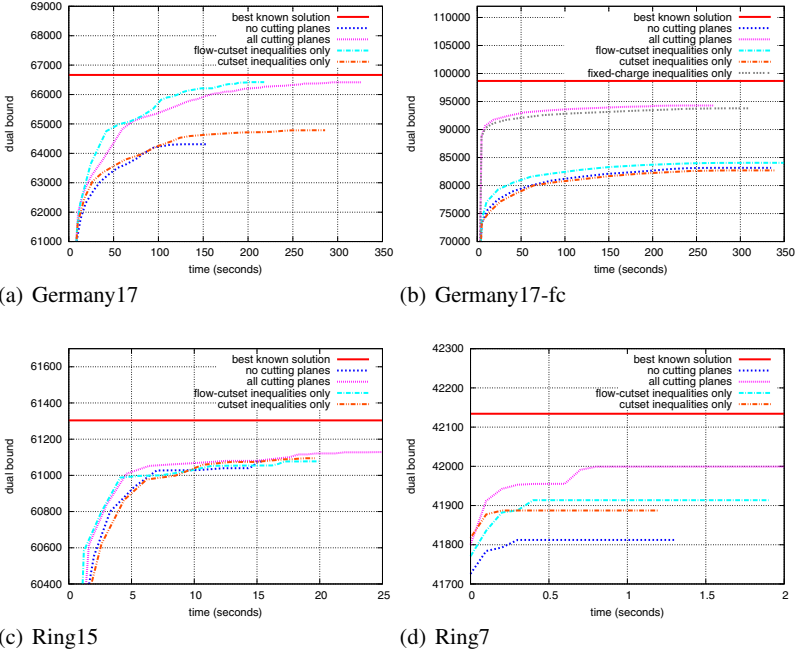
(a) Germany17

(b) Germany17-fc

(c) Ring15

(d) Ring7

Fig. 3.2 Unprotected demands: dual bound at the root node

graph (V, L) contains edges between almost all node pairs, whereas only a few links cross a cut in single-layer graphs. Further, we have implemented our cutting planes as callbacks in SCIP, whereas in [28], CPLEX was used as a branch-and-cut framework, which means that different general-purpose cutting planes have been used.

For the problem Germany17-fc with physical cost, most of the optimality gap comes from the z_e variables whose values are highly fractional and close to 0 in the solution of the LP relaxation. A major part of this gap is closed by the fixed-charge inequalities that operate on the physical layer. Of course, the contribution of these inequalities changes with the ratio of the costs of the physical fiber links to the logical wavelength links and the node hardware.

In contrast to these three instances, the problem-specific cutting planes have only a marginal effect on the dual bound for Ring15 compared to that of SCIP's built-in general-purpose cuts. This is probably due to the fact that in SCIP's default settings, the dual bound at the end of the root node is within 0.4 % of the optimal solution value, so there is not much room for improvement at all. We also observed that on this instance, our cuts seem to interfere with the c-mir and Gomory cuts separated by SCIP, which are based on a mixed-integer rounding procedure similar to the one described in Section 3.4. With these cuts disabled in SCIP, our inequalities could reduce the relative distance between the root dual bound and the best-known solution from 3.8 % to 0.4 %, thus achieving the same dual bound as that of SCIP's

cutting planes. The number of violated cutting planes found in this setting is reported in Table 3.2 for all instances.

Table 3.2 Number of violated cutset (3.3), flow-cutset (3.5), and fixed-charge inequalities (3.6) found in root of branch-and-bound tree without separation of SCIP built-in cuts

instance	# cuts unprotected			# cuts protected		
	cutset	flow-cutset	fixed-charge	cutset	flow-cutset	fixed-charge
Germany17	37	1521	-	4	940	-
Germany17-fc	34	1046	35	7	844	20
Ring15	66	652	-	26	489	-
Ring7	41	98	-	15	24	-
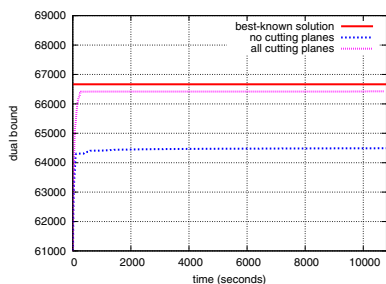
(a) Germany17   (b) Germany17-fc
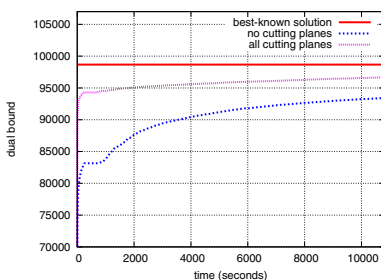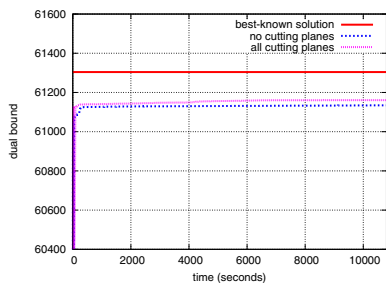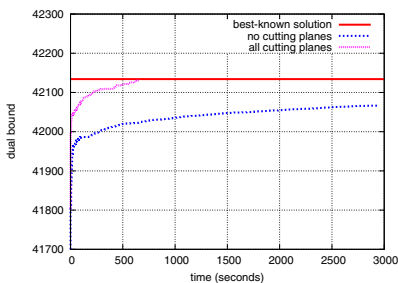
(c) Ring15   (d) Ring7

Fig. 3.3 Unprotected demands: dual bound during test runs of three hours

In a second study, we have investigated the lasting effect of the cutting planes on the dual bound in longer computations. Figure 3.3 shows the development of the dual bound with and without all cutting planes from Section 3.4 during a computation with a time limit of three hours for all four test instances, compared to the best-known solution. Similarly to most of SCIP's own cutting planes, we separated our inequalities only at the root node of the branch-and-cut tree.

By applying all separators we could solve the problem Ring7 to optimality within ten minutes, whereas without our cutting planes the computation was aborted after nearly one hour with a nonzero optimality gap due to the memory limit of 2 GB. The size of the search tree was 1.2 million unexplored nodes at this point (and four million explored nodes). Figure 3.3 shows that the dual bounds obtained with our cutting planes are very close to their maximum possible values. In fact, as the upper bound improved in both cases, the relative gap between the dual bound and the best solution found in that specific run (as opposed to the best solution known) could be improved from 4 % to 0.36 % and from 12.4 % to 3.1 %, respectively. For Ring15 the improvement of the dual bound by the cutting planes was much smaller than that for the other instances, probably for the reasons discussed above.

3.5.3 Protected Demands



(a) Germany17
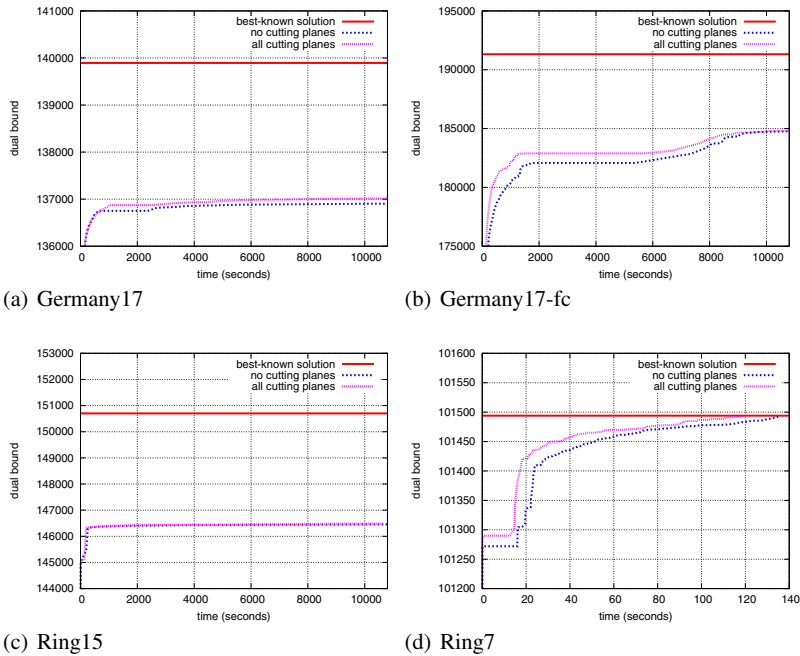
(b) Germany17-fc

(c) Ring15

(d) Ring7

Fig. 3.4 Protected demands: lower bound in test runs of three hours

In the case of protected demands, we first of all would like to point out that the problem size drastically increases compared to the unprotected case. Instead of $|V| - 1$ commodities, $|H|$ commodities have to be routed, increasing the number of

variables and constraints considerably. Consequently, solving the initial LP relaxation, as well as reoptimizing the LP after adding a cutting plane or a branching constraint, takes more time with protection than without.

With 1+1 protected demands, the cutting planes have only a marginal effect on the dual bound. Figure 3.4 shows the increase of the dual bound in a three-hour test run with and without cutting planes (again, the solid red line at the top indicates the best-known solution value). It can be seen that the dual bound always increases, but only by a very limited amount. More detailed investigations revealed that the small progress is mainly due to the strength of the general-purpose c-mir and Gomory cuts generated by SCIP. Experiments where these cuts were turned off showed that our inequalities still contribute significantly to closing the optimality gap at the root node. Table 3.2 shows the number of violated inequalities found at the root node in this setting. Only slightly lower numbers of violated inequalities are found with c-mir and Gomory cuts turned on, but their impact on the dual bound is limited in such a case; cf. Figure 3.4.

3.5.4 Preprocessing and Heuristics

(a) Unprotected instances   (b) Protected instances
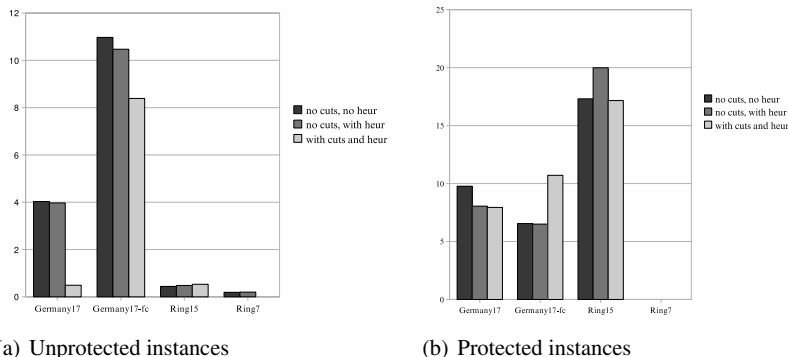
Fig. 3.5 Optimality gaps after three hours without cuts and heuristics; with heuristics only; and with both cuts and heuristics

We also tested the combined effect of our primal heuristics and cutting planes on the optimality gap after three hours. Figure 3.5 shows these gaps for each of the networks in three settings: without cuts and heuristics; with heuristics; and with both heuristics and cuts. The protected Ring7 network has no bars because it was solved to optimality in all cases; we will discuss this network below.

In five out of the seven other instances, adding our cutting planes reduced the optimality gaps. There were two exceptions: On the protected Germany17-fc network, separating the cuts at the root nodes took so much time that a significantly smaller

number of branch-and-cut nodes could be solved within the time limit, leading to a much worse upper bound and a slightly worse lower bound. On the unprotected Ring15 network, the final dual bound was better with cuts than without, but the primal bound was a bit worse, leading to a slightly larger gap. From a practical point of view, however, the difference is marginal.

The effect of our primal heuristics is similar. On five out of the eight instances, the heuristics helped to reduce the optimality gap or the time needed to solve the problem to optimality, and in one instance (unprotected Ring7) there was no difference. In fact, our heuristics found the best solution after three hours in nine out of the 16 cases where they were called; in four of them, the best solution was found by REROUTINGMIP at the root node. Also, on the two instances Ring15 (unprotected) and Germany17-fc (protected), the heuristics found improving solutions early in the branch-and-cut tree, but the resulting traversal of the search tree led to a worse primal bound after the fixed time limit of three hours than without the heuristics. Unfortunately, such effects are rather unpredictable.
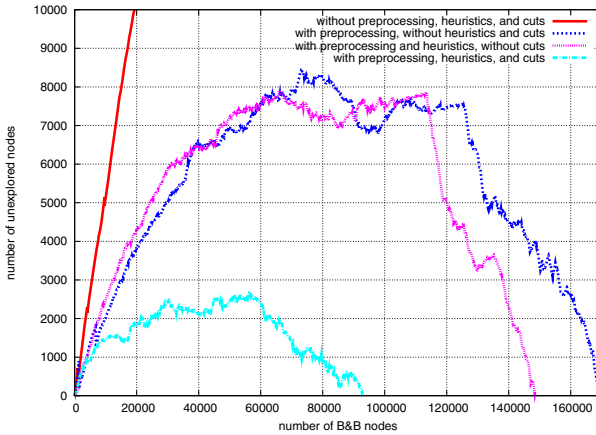
Fig. 3.6 Number of unexplored branch-and-cut nodes on the protected Ring7 network

For the protected Ring7 network, Figure 3.6 shows that the maximum number of unexplored nodes in the search tree was roughly reduced by 2/3 by our cutting planes, even though they were added only in the root node. Also, the heuristics helped in getting a smaller search tree by finding good solutions early in the search tree. The GROOMCAPHEUR heuristic found an optimal solution after 353 nodes, compared to 6,825 nodes without the heuristics. This caused large parts of the search tree to be cut off. A separate run where cutting planes, heuristics, and preprocessing were switched off is shown in the fourth curve at the top of Figure 3.6. It can be seen that the preprocessing also significantly helped to reduce the size of the search tree. With all our plug-ins disabled, an optimal solution was found only after 9,626 nodes, and the size of the search tree grew to more than 780,000 unexplored nodes because of the weak lower bound.

3.6 Conclusions

In this work, we have presented a mixed-integer programming model for a two-layer SDH/WDM network design scenario. The model includes many practically relevant side constraints such as many parallel logical links, various bit rates, node capacities, and survivability with respect to physical node and link failures. To accelerate the solution process for this planning task, we have applied problem-specific preprocessing, a variety of network design-specific cutting planes, and MIP-based primal heuristics within the branch-and-cut framework SCIP. These ingredients have been tested on several realistic planning scenarios provided by Nokia Siemens Networks.

With unprotected demands, our cutting planes significantly raised the lower bounds to close to the optimal solution value. With 1+1 protection against physical failures, they also helped to improve the dual bounds, but less than in the unprotected case. The preprocessing steps, although relatively simple, turned out to be crucial for reducing the size of the branch-and-cut tree. Although the effect of the MIP-based heuristics was not so clear, they found the optimal solution early in the search tree in several instances, sometimes even at the root node. The fact that these heuristics can easily be generalized to other network design problems and side constraints makes the sub-MIP approach very flexible.

Although the presented methods could significantly reduce the computation times for the considered realistic networks, they can still be improved. First, the presented methods do not scale well with the network size because the edge-flow formulation gets too large. Second, fast combinatorial routing heuristics have to be developed in addition to the MIP-based heuristics in order to find good survivable routings that can be used in primal solutions. Third, cutting planes are needed that better take the inter-layer dependencies into account.

References

1. Achterberg, T.: Constraint Integer Programming. Ph.D. thesis, Technische Universität Berlin (2007). http://opus.kobv.de/tuberlin/volltexte/2007/1611/
2. Achterberg, T.: SCIP: solving constraint integer programs. Mathematical Programming Computation **1**(1), 1–41 (2009). URL http://scip.zib.de/
3. Alevras, D., Grötschel, M., Wessäly, R.: A network dimensioning tool. ZIB Technical Report SC-96-49, Konrad-Zuse-Zentrum für Informationstechnik Berlin (1996)
4. Atamtürk, A.: On capacitated network design cut-set polyhedra. Mathematical Programming **92**, 425–437 (2002)

5. Baier, G., Engel, T., Autenrieth, A., Leisching, P.: Mehrperiodenplanung optischer Transport-netze. In: 7. ITG-Fachtagung Photonische Netze, Leipzig, Germany, vol. 193, pp. 153–160. VDE-Verlag (2006)

6. Belotti, P., Capone, A., Carello, G., Malucelli, F., Senaldi, F., Totaro, A.: Design of multi-layer networks with traffic grooming and statistical multiplexing. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007)

7. Bienstock, D., Chopra, S., Günlük, O., Tsai, C.: Mininum cost capacity installation for multi-commodity flows. Mathematical Programming **81**, 177–199 (1998)

8. Bienstock, D., Günlük, O.: Capacitated network design – polyhedral structure and computa-tion. INFORMS Journal on Computing **8**(3), 243–259 (1996)

9. Chopra, S.: On the spanning tree polyhedron. Operations Research Letters **8**, 25–29 (1989)

10. Chopra, S., Gilboa, I., Sastry, S. T.: Source sink flows with capacity installation in batches. Discrete Applied Mathematics **86**, 165–192 (1998)

11. Dahl, G., Martin, A., Stoer, M.: Routing through virtual paths in layered telecommunication networks. Operations Research **47**(5), 693–702 (1999)

12. Dawande, M., Gupta, R., Naranpanawe, S., Sriskandarajah, C.: A traffic-grooming algorithm for wavelength-routed optical networks. INFORMS Journal on Computing **19**(4), 565–574 (2007).

13. Fortz, B., Poss, M.: An improved Benders decomposition applied to a multi-layer network design problem (2008). Optimization Online preprint 1919

14. Günlük, O.: A branch-and-cut algorithm for capacitated network design problems. Mathemat-ical Programming **86**, 17–39 (1999)

15. Höller, H., Voss, S.: A heuristic approach for combined equipment-planning and routing in multi-layer SDH/WDM networks. European Journal of Operational Research **171**(3), 787–796 (2006)

16. ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA: CPLEX 10.1 Reference Manual (2006). URL http://www.cplex.com

17. Knippel, A., Lardeux, B.: The multi-layered network design problem. European Journal of Operational Research **183**(1), 87–99 (2007)

18. Koster, A. M. C. A., Orlowski, S., Raack, C., Baier, G., Engel, T.: Single-layer cuts for multi-layer network design problems. In: Telecommunications Modeling, Policy, and Tech-nology, chap. 1, pp. 1–23. Springer (2008). URL http://dx.doi.org/10.1007/978-0-387-77780-1_1. Selected proceedings 9th INFORMS Telecommunications Conference

19. Koster, A. M. C. A., Zymolka, A.: Minimum converter wavelength assignment in all-optical networks. In: Proceedings of ONDM 2004, pp. 517–535. The 8th IFIP Working Conference on Optical Network Design and Modelling, Ghent, Belgium (2004)

20. Koster, A. M. C. A., Zymolka, A.: Tight LP-based lower bounds for wavelength conversion in optical networks. Statistica Neerlandica **61**(1), 115–136 (2007)

21. Kubilinskas, E., Pióro, M.: An IP/MPLS over WDM network design problem. In: Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal, vol. 3, pp. 718–725 (2005)

22. Magnanti, T. L., Mirchandani, P.: Shortest paths, single origin-destination network design and associated polyhedra. Networks **33**, 103–121 (1993)

23. Marchand, H., Wolsey, L. A.: Aggregation and mixed integer rounding to solve MIPs. Oper-ations Research **49**(3), 363–371 (2001)

24. Nemhauser, G., Wolsey, L. A.: Integer and Combinatorial Optimization. John Wiley & Sons (1988)

25. Orlowski, S., Koster, A. M. C. A., Raack, C., Wessäly, R.: Two-layer network design by branch-and-cut featuring MIP-based heuristics. In: Proceedings of the 3rd International Net-work Optimization Conference (INOC 2007), Spa, Belgium (2007)

26. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0–Survivable Network De-sign Library. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007). http://sndlib.zib.de

27. Ortega, F., Wolsey, L. A.: A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. Networks **41**, 143–158 (2003)
28. Raack, C., Koster, A. M. C. A., Orlowski, S., Wessäly, R.: On cut-based inequalities for capacitated network design polyhedra. Networks (2009). Submitted for publication, available in parts as ZIB Reports 07-08 and 07-14, Konrad-Zuse-Zentrum für Informationstechnik Berlin
29. Raghavan, S., Stanojević, D.: WDM optical design using branch-and-price (2007). Robert H. Smith School of Business, University of Maryland
30. Savelsbergh, M.: Preprocessing and probing for mixed integer programming problems. ORSA Journal on Computing **6**, 445–454 (1994)
31. Wolsey, L. A.: Integer Programming. John Wiley & Sons (1998)

Chapter 4
Routing and Label Space Reduction in Label Switching Networks

Fernando Solano, Luis Fernando Caro, Thomas Stidsen, and Dimitri Papadimitriou

Abstract This chapter is devoted to the analysis and modeling of some problems related to the optimal usage of the label space in label switching networks. Label space problems concerning three different technologies and architectures – namely Multi-protocol Label Switching (MPLS), Ethernet VLAN-Label Switching (ELS) and All-Optical Label Switching (AOLS) – are discussed in this chapter. Each of these cases yields to different constraints of the general label space reduction problem. We propose a generic optimization model and, then, we describe some adaptations aiming at modeling each particular case. Simulation results are briefly discussed at the end of this chapter.

Key words: MPLS, AOLS, ELS, label space reduction

4.1 Introduction to Label Switching

Label switching architectures base their forwarding decisions in contiguous fixed length identifiers referred as labels. In these architectures packets are forwarded

Fernando Solano
Institute of Telecommunications, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland, e-mail: `fs@tele.pw.edu.pl`

Luis Fernando Caro
Institute of Informatics and Applications, University of Girona, Campus Montilivi, 17071 Girona, Spain, e-mail: `lfcaro@atc.udg.edu`

Thomas Stidsen
Informatics and Mathematical Modeling, Danish Technical University, Richard Petersens Plads, DK-2800 Kgs. Lyngby, Denmark, e-mail: `tks@imm.dtu.dk`

Dimitri Papadimitriou
Network and Technology Strategy, Alcatel-Lucent, Copernicuslaan 50, B-2018 Antwerpen, Belgium, e-mail: `dimitri.papadimitriou@alcatel-lucent.be`

through specific sequences of nodes, named Label Switched Paths (LSPs) [10]. For this purpose each packet is marked with a label depending on the LSP through which it is sent. Each node uses Label as the index to look up in the forwarding table both the node where the packet needs to be forwarded and the new label used to identify the same packet in the next node.

We refer to a *label space* as the set of available labels that can be bound to paths (flows). There are many ways in which a label space can be configured [7]. In this chapter, we treat only two of them: *label space per node* and *label space per link* (or interface). In a label space per node, every single LSP must be identified with a different label *regardless* of its incoming interface. In contrast, a label space per interface may assign the same label to two different paths if they have different incoming interfaces.

A packet header may contain not only a single label, but a stack of labels. In order to handle properly this stack, there are three operations that can be performed at every hop:

- Label SWAP: The incoming packet's top Label value is replaced by the outgoing Label value and the packet is switched to an interface towards the next node.
- Label SWAP and PUSH: The incoming label of a packet is swapped, and then one or more labels are stacked, leaving the new labels on the top of the stack.
- Label POP: The top label of the packet is removed from the stack.

Depending on the specific underlying protocols and hardware architecture, all or some of the operations may be supported. Each label switching architecture has its own motivations and objectives for using these techniques, as we will explain in the following section.

There are several methods that use the three label operations to allow different LSPs sharing a label at an intermediate node, thus reducing the number of labels in the forwarding table. Each label switching architecture has its own motivations and objectives for using these techniques. Based on this, the label space reduction problem consists of the efficient usage of the label spaces considering the limitations of a given particular technology (e.g., supported operations, label space configuration, label space size).

In this chapter we study the problem of minimizing the number of labels used in networks using three different label switching architectures. Since each of the technologies is aimed at being used for different purposes, the objectives and restrictions of the problem become different in each case, while preserving the main structure of the solution. Therefore, in Section 4.2, we give an outline of the most important differentiating aspects of these three technologies from the point of view of the aforementioned problem. In Section 4.3, we present two methods that can be used for reducing the number of labels used in a network. In Section 4.4, we explain how considering the problem of finding the routes together with reducing the usage of the label space makes the problem harder to solve. In Section 4.5, we present an Integer Linear Program (ILP) formulation for the most generic problem. Later, in the same section, we present some extensions to the formulation that model each particular previously described case. In Section 4.6, we depict simulation results for

the problems related to each of the three technologies. A summary and conclusions of the chapter are given in Section 4.7.

4.2 Functional Description of the Technologies

The *label space reduction* problem has been mainly studied for three different label switching architectures: Multi-protocol Label Switching Traffic Engineering (MPLS-TE), All-Optical Label Swapping (AOLS), and Ethernet VLAN-Label Switching (ELS). We give an overview of the technologies and their particular motivation for reducing label spaces in subsequent subsections.

4.2.1 Multi-protocol Label Switching Traffic Engineering (MPLS-TE)

MPLS-TE is a technology designed to implement label switching with constraint-based routing. MPLS-TE uses a 20-bit label and the Resource Reservation Protocol for Traffic Engineering (RSVP-TE) for signaling LSPs [1].

The RSVP-TE protocol working principle is based on soft states. A soft state is a variable in memory that stores all the necessary information about the flow and its characteristics. RSVP-TE stores one soft state per path (flow) and, due to its nature, soft states must be refreshed periodically.

RSVP-TE scalability properties are bounded as follows:

1. In order to provide QoS, resource reservation, resilience, etc., the amount of information stored per state is considerably large (resulting in memory consumption),
2. Due to its soft state nature, RSVP-TE needs state refreshing, resulting in

   - bandwidth usage for the transmission of refreshing messages, and
   - CPU processing upon the arrival of a refreshing message from neighbors.

Since MPLS was designed for simplifying forwarding, in its most basic form, labels are the only information that is extracted (and considered) from the packet at forwarding. Therefore, when considering RSVP-TE, the relationship between labels and soft states is one-to-one . Otherwise, either two flows (using different labels) would share the same reservation, or a node would not be able to distinguish for one flow the correct reservation state. Throughout this chapter, we study the label space reduction problem with the main objective of reducing the number of needed soft states by RSVP-TE.

4.2.2 All-Optical Label Switching (AOLS)

AOLS is the name given to a set of technology proposals aiming at performing packet label switching using purely optical signals. We consider the proposal made in the LASAGNE [9] project since it best maps the functionality of MPLS-TE packet label switching in this sense. Similarly to MPLS-TE, the LASAGNE proposal also offers label spaces per link and per interface [10].

Even though the LASAGNE project achieves its objective, it is not scalable since it requires a special optical device for each label used . Therefore, a cost-efficient implementation of this technology requires a significant reduction in the usage of the label spaces as much as is possible.

Label stripping refers to a technique that encodes the route of an LSP in the stack, so at every hop the pertinent LSR strips off (pops) the top label and determines the next hop based on its content. In label stripping, labels are never swapped or pushed at core nodes. In other words, label stripping encodes the route of the path in the header. Therefore, since all the paths use one label for every hop, every node v_i must store at most $\Delta(v_i)$ labels, where $\Delta(v_i)$ is the degree of the node $v_i \in V$. Clearly, the number of labels that must be encoded in the stack (henceforth the stack size) is equal to the number of hops of the route.

The label stripping [2] strategy for label switching yields to a lower bound on the label space usage: one label per link regardless of the number of LSPs. However, it increases the stack size to the maximum length of any path in consideration, wasting more bandwidth due to the need of a larger space for optical header encodings. Therefore, better label switching strategies with less drastic trade-offs are desired.

4.2.3 Ethernet VLAN-Label Switching (ELS)

In [8] and [4] a scheme that enables an Ethernet network to create an LSP is discussed. The label is encoded in the 12-bit S-VID tag field of the 802.1ad frame. Push and pop operations are only suported at the source and destination nodes. Label swapping is implemented by using the S-VID translation operation defined in IEEE 802.1ad. Figure 4.1 describes the label operations along an Ethernet LSP.

Given that ELS supports label swapping and uses a 12-bit S-VID Tag field as the label, labels must be unique only in the context of a network link (labels have link scope). Therefore, in ELS there is a limit of 4,096 (2^{12}) LSPs per link in the network. Given that MPLS-TE has a limit of 1,048,576 (2^{20}) LSPs per link and that label stacking is not supported, ELS might be affected by label scalability issues, in the sense that an LSP request can be blocked due to unavailability of labels instead of bandwidth . The label space problem in this technology is to reduce the maximum number of labels used so it does not exceed the 12-bit limit.
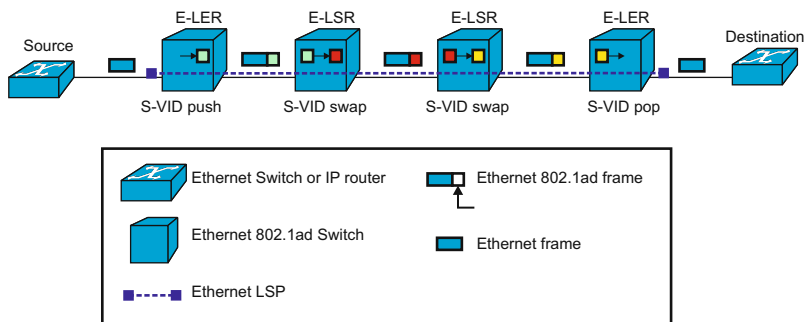
Fig. 4.1 Ethernet VLAN-Label Switching LSP example

4.3 Methods for Scaling the Usage of the Label Space

In this section we summarize the two methods considered in the literature for reducing label spaces in label switching technologies.

4.3.1 Label Merging

Label merging uses the label swapping operation to assign the same label to two or more LSPs in a continuous and common segment that goes from any common intermediate node to the same destination node. All LSPs must follow the same path from the intermediate node to the destination node in order to be merged. Label merging is able to reduce the number of labels used in a link. It can be used in label switched networks where nodes are capable of performing label swapping (MPLS-TE, AOLS and ELS). An example is presented in Figure 4.2, where label merging between the two LSPs is possible at link $(N5, N6)$ and not at link $(N1, N2)$.

A single label is assigned to a set of LSPs in a link if either they have the same label in the following downstream link and the downstream link is the same, or the link is the last link for both LSPs. When label merging is applied, assigning the labels to a set of established LSPs is trivial. The problem can be solved in polynomial time, guaranteeing the optimal assignment in terms of labels used. This has been discussed in [15].

If we consider all the paths that are forwarded in one link, it is possible to group them according to the label they would be using, taking into account the label merging method. In the following sections we will use the term Merging Link (MERLIN) group of Paths for each of these partitions for a given link.
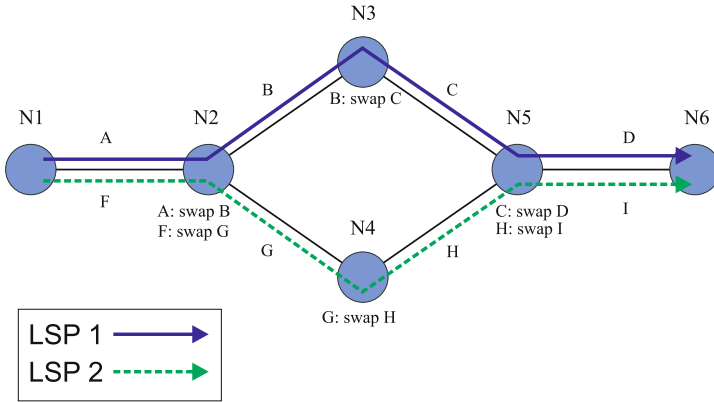
(a) No Merging
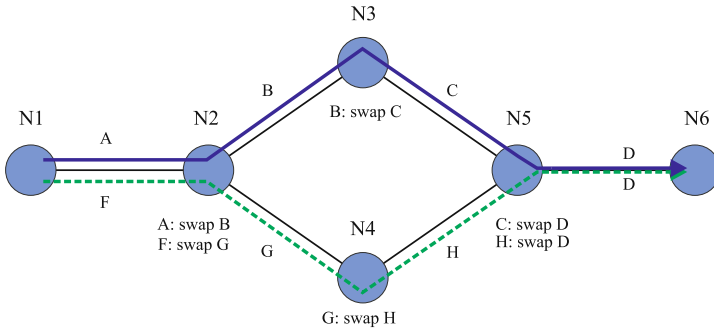
(b) Merging

Fig. 4.2 Label merging example

4.3.2 Label Stacking

As mentioned before, a packet may contain one or more labels in its stack. The stack can be used in order to create a hierarchy of LSPs [6]. In this sense, a 'bigger' LSP covers a group of 'smaller' LSPs, as seen in Figure 4.2 (a). Henceforth, we name the bigger LSPs covering the smaller ones *tunnels*. In this chapter, we consider only two hierarchies of LSPs. Larger hierarchies are left for further analysis. Even though a LSP can 'join' a tunnel at any point, it must 'leave' the tunnel at the end, creating an asymmetry in the tunnel.

This method allow us to reduce the label space since, instead of using one label per hop per LSP, we use one label per hop per tunnel regardless of how many LSPs are covered. Let us suppose that we have a tunnel t of length $l(t) > 1$ hops,[1] and it covers a set of LSPs $P_t = \{p_0, p_1, \ldots, p_{|L_t|}\}$. Let us assume that the tunnel t covers

---

[1] Tunnels of 1 hop length are discouraged since, due to technological reasons, they increase the label space by 1 in all cases.

LSP p_i in $l(p_i) \leq l(t)$ hops. Then we can compute the reduction in the label space reduction incurred by this tunnel as

$$r(t) = \sum_{p_i \in P_t} (l(p_i) - 1) - (l(t) - 1)$$
$$= \sum_{p_i \in P_t} l(p_i) - l(t) + 1 - |L_t|.$$

Given a set of LSPs $\mathscr{P}$, we can define the LABEL STACKING problem as how to create a set of tunnels (covering the given LSPs) in such a way that we reduce the overall label space as much as possible. It is not necessary to cover all the links of all the LSPs; however, the more LSP hops are covered, the greater the reduction could be. Obviously, a path cannot be covered by more than one tunnel in any hop.

A solution to the problem is given by the set of tunnels used, $\mathscr{T}^*$, and a proper mapping from LSPs $\mathscr{P}$ to tunnels $\mathscr{T}^*$, i.e., $\mathscr{P} \to \mathscr{T}^*$. For instance, Figure 4.3 shows two solutions to the same problem achieving different label space reductions. Even though the problem has not been formally proved to be NP-complete, several algorithms [11–13] have been proposed to find an optimal solution, yet without guaranteeing it.
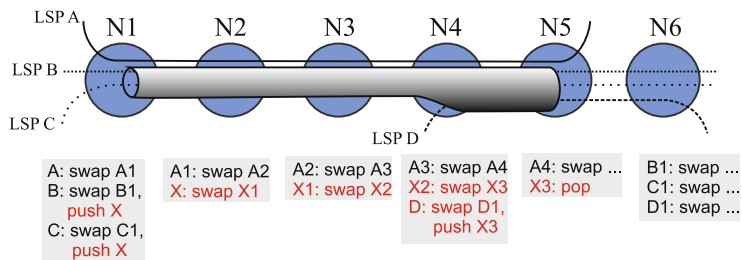
(a) Suboptimal solution
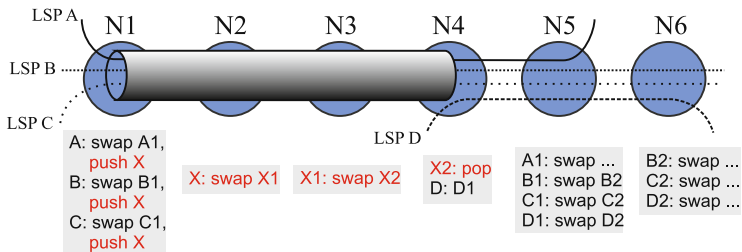
(b) Optimal solution

Fig. 4.3 Stacking problem example

It is worth highlighting the following theorem:

Theorem 4.1 (Solano et al. [16]). *Given a set of paths, a set of tunnels* $\mathcal{T} \supseteq \mathcal{T}^*$ *can be computed in polynomial time, and this set contains the tunnels considered in the optimal solution* $\mathcal{T}^*$ *for the* LABEL STACKING *problem.*

It is worth noticing that $|T|$ belongs to $\mathcal{O}(l^3 \cdot n)$, with l representing the number of LSPs and n the number of nodes. Yet, selecting the proper tunnels $\mathcal{T}^*$ in set $\mathcal{T}$ is not trivial. Furthermore, given the proper tunnels $\mathcal{T}^*$, deciding which LSPs should be covered by each tunnel (the mappings $\mathcal{P} \to \mathcal{T}^*$) is also not trivial.

We will consider henceforth that reducing the amount of labels used can be achieved using both label merging and stacking methods together for the same set of LSPs [14]. In this sense, it must be taken into account that labels can be merged if they belong to the same hierarchy, e.g., merging labels used for path (tunnel) identification with only those used for path (tunnel) identification (see Figure 4.4).
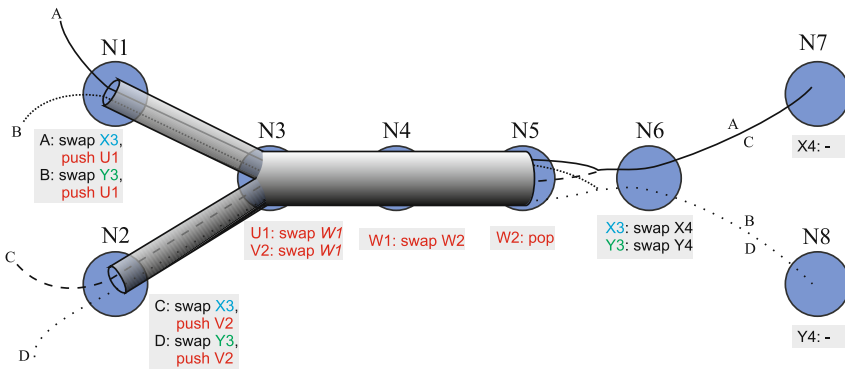
Fig. 4.4 Label merging at both hierarchies

4.4 Considering Routing

Previously, we have assumed that the path routes are given. However, for some technologies (e.g., AOLS and ELS), the label space is too tightly limited, or very expensive, and selection of the path routes might help in the label space reduction.

When considering the use of label merging and/or label stacking to reduce label space, there is a trade-off. The trade-off is presented because these techniques give higher reduction when paths are aggregated (sharing link capacity); however, this incurs at the same time a higher bandwidth consumption. The reason is that the more hops a set of paths shares, the more the profit can be obtained when a tunnel is placed over them. In this way, a routing algorithm designed to utilize the minimum number of labels would aim at placing the path demands over the same set of links when possible. In this case, a routing algorithm designed to minimize link overutilization would do the opposite.

In Figure 4.5 we show an example. Nodes $N1$ and $N2$ are ingress and nodes $N6$ and $N7$ are egress. We set the link capacity to two units of traffic. Considering the topology in Figure 4.5 (a) and that we want to route one unit of traffic between each pair of ingress-egress nodes, a routing algorithm minimizing the link overutilization would route the traffic as in Figure 4.5 (b).
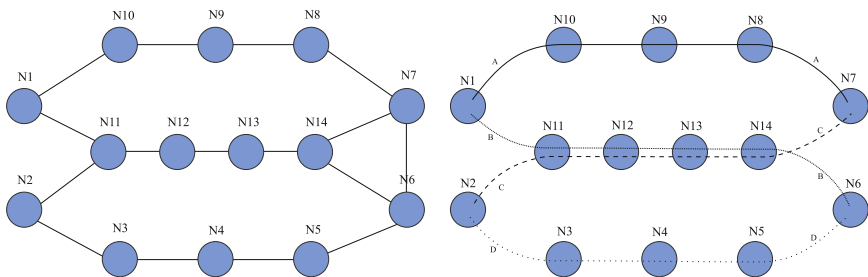
(a) Network Topology. Ingress nodes are $N1$ and $N2$. Egress nodes are $N6$ and $N7$

(b) Routing solution optimizing maximum link utilization, using 22 labels

Fig. 4.5 Example scenario

In Figure 4.6 (a), we show the Minimum Interference Routing Algorithm (MIRA) routing solution [5] when applying the merging and stacking techniques. The number of labels used is 20. However, in Figure 4.6 (b), we show a routing solution, without minimizing link overutilization, that uses four labels less.
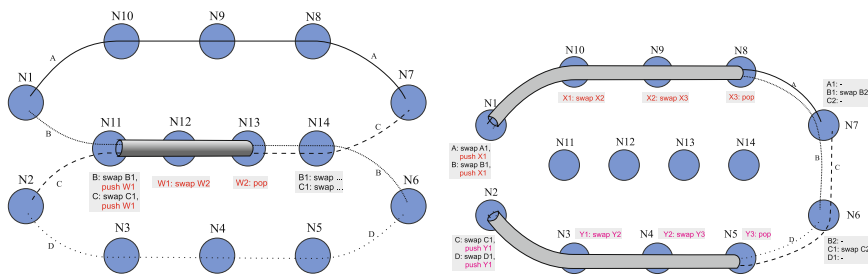
(a) Solution without label stacking using 20 labels

(b) Solution with label stacking using 18 labels

Fig. 4.6 Solution to the problem considering a limit of two traffic units in the links capacity

When LSPs are being aggregated, increasing the link bandwidth can reduce link overutilization and even reduce the number of labels used. This is illustrated in Figure 4.7, where by increasing the bandwidth of the links between nodes $N11$ and $N14$ to four units of traffic, the number of labels is less than those in the previous solutions.

(a) Solution with four traffic units as link capac- (b) Solution with four traffic units as link capac-
ity and no label stacking using 16 labels   ity and label stacking using 12 labels
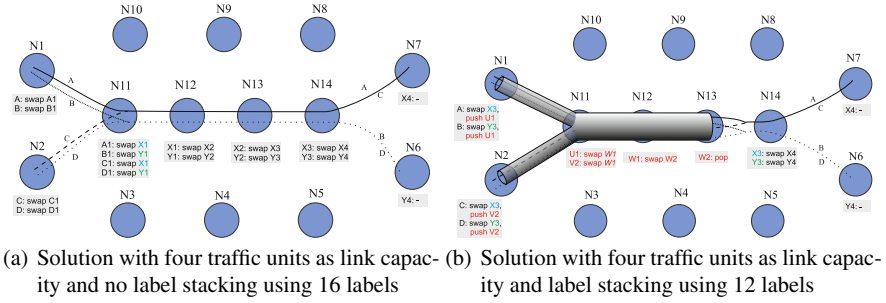
Fig. 4.7 Solution to the problem considering a limit of four traffic units in the link capacity

At this point, the network administrator is in charge of measuring whether the label space reduction compensates the (extra) allocated bandwidth.

4.5 Generic Model

In this section we propose a generic model for the ROUTING AND LABEL SPACE REDUCTION problem, in which given a set of demands we are asked to route them, constrained by a link capacity, with the objective of minimizing the number of labels used. In subsequent subsections, we give other formulations in order to consider similar problems.

The ILP model proposed is path-based. Therefore, all feasible paths in the network are initially generated using an exponential running time algorithm. Our routing solution would consist then of selecting a subset of these paths.

Considering the computed paths, all the feasible tunnels are computed as mentioned in Theorem 4.1.

4.5.1 Parameters and Variables

The following is a list of all the indexes used in the model.

- $i, j \in V$ represent nodes in the network.
- $(i, j) \in E \subseteq V \times V$ represents a link in the network.
- $\alpha \in \mathscr{P}$ represents a generated path in the network.
- $\phi \in T$ represents a tunnel in the network.
- $m, n \in \mathbb{N}$ represents a MERLIN group identifier.

The parameters used in the model are the following.

- S_i^α is set to 1 if node i is the source of path α.
- D_i^α is set to 1 if node i is the destination of path α.

- $C_{i,j}$ is set to the total bandwidth demand between nodes i and j.
- κ is set to the minimum of the desired Maximum Link Utilization (MLU) and the link capacity.

In addition, two parameters can be seen as functions in the model.

- $P^m_{(i,j)}$ is the function that, given a path MERLIN group m and a link (i,j), evaluates to the set of paths belonging to the group.
- $T^n_{(i,j)}$ is the function, which, given a tunnel MERLIN group n and a link (i,j), evaluates to the set of tunnels belonging to the group.

The variables used in the model are the following.

- $\bar{x}^\alpha$ is set to 1 when path α is used to route any demand.
- x^α is set to the bandwidth allocated on path α.
- $y^{\phi,\alpha}$ is set to 1 when path α is covered by tunnel ϕ.
- $z^m_{(i,j)}$ is set to 1 when the path MERLIN group m uses a label on link (i,j).
- $\hat{z}^n_{(i,j)}$ is set to 1 when the tunnel MERLIN group n uses a label on link (i,j).

4.5.2 Integer Linear Program for the Network Design Problem

The objective function is minimizing the overall number of labels used (or MERLIN groups) in the network (for both paths and tunnels):

$$\min \sum_{(i,j)\in E} \left(\sum_{m\in\mathbb{N}} z^m_{(i,j)} + \sum_{n\in\mathbb{N}} \hat{z}^n_{(i,j)} \right) \tag{4.1a}$$

$$\text{s.t.} \sum_{\alpha:S^\alpha_i=D^\alpha_j=1} x^\alpha \geq C_{i,j}, \qquad \forall i,j \in V, i\neq j \tag{4.1b}$$

$$\sum_{\alpha|\exists m,\alpha\in P^m_{(i,j)}} x^\alpha \leq \kappa, \qquad \forall i,j \in V, i\neq j \tag{4.1c}$$

$$\kappa\cdot\bar{x}^\alpha - x^\alpha \geq 0, \qquad \forall\alpha\in\mathscr{P} \tag{4.1d}$$

$$\bar{x}^\alpha - \sum_{\phi:(i,j)\in\phi\cap\alpha} y^{\phi,\alpha} \geq 0, \qquad \forall(i,j)\in E, \alpha\in\mathscr{P} \tag{4.1e}$$

$$K\cdot\hat{z}^n_{(i,j)} - \sum_{\phi\in T^n_{(i,j)},(i,j)\in\phi\cap\alpha} y^{\phi,\alpha} \geq 0, \qquad \forall(i,j)\in E, n\in\mathbb{N} \tag{4.1f}$$

$$K\cdot z^m_{(i,j)} - \sum_{\alpha\in P^m_{(i,j)}} \left(\bar{x}^\alpha - \sum_{\phi:(i,j)\in\alpha\cap\phi,D^\phi_j=0} y^{\phi,\alpha} \right) \geq 0, \quad \forall(i,j)\in E, m\in\mathbb{N} \tag{4.1g}$$

Inequalities (4.1b) assure that all traffic is routed. Inequalities (4.1c) limit the capacity that can be used in every link to κ units of traffic. Each of the inequalities in (4.1d) sets a path requiring labels if it is being used by some demand.

Inequalities (4.1e) relate the variables $\bar{x}^{\alpha}$ with $y^{\phi,\alpha}$. Since $\bar{x}^{\alpha}$ is binary, each inequality states that at most one tunnel can be used for a path at every link. The constant K is set to the minimum number such that $K > \sum_{m\in\mathbb{N}} \left| P^m_{(i,j)} \right|, \forall i,j \in V$.

In inequalities (4.1f), the variable $\hat{z}^n_{(i,j)}$ pays (i.e., is set to 1) for the use of one label for all the active tunnels ϕ intersecting any active path α. Inequalities (4.1g) work similarly, but they are concerned with paths merging. They differ from (4.1f) in that the term $\sum_{\phi} y^{\phi,\alpha}$ is added in order to avoid paying for paths that have been covered.

4.5.3 Traffic Engineering Formulation

The formulation previously shown is appropriate for a network design problem, in which we want to plan how many labels we need for a given traffic demand matrix. In the following, we show how to modify it in order to solve traffic engineering problems. In a TE problem, we are given a traffic demand matrix and we are asked to maximize the throughput given bounds on link capacities and label space sizes.

We introduce a new variable $w_{i,j}$ which is set to 1 when the traffic from i to j has been routed. Therefore, our new objective function is

$$\max \sum_{i,j\in V} C_{i,j} \cdot w_{i,j} \tag{4.2}$$

In addition, inequalities (4.1b) are relaxed in the following way:

$$\sum_{\alpha:S^{\alpha}_i=D^{\alpha}_j=1} x^{\alpha} - C_{i,j} \cdot w_{i,j} \geq 0, \forall i,j \in V \tag{4.3}$$

The formulation in this form becomes unbounded. We proceed to bound it according to the way in which the label space is configured.

4.5.3.1 Bounded Label Space per Node

Bounding the label space of a node to a maximum size can be done by

$$\sum_{i\in V, m,n\in\mathbb{N}} \left(z^m_{(i,j)} + \hat{z}^n_{(i,j)} \right) \leq L_j, \forall j \in V \tag{4.4}$$

with L_j being the maximum number of labels *per node* at j.

4.5.3.2 Bounded Label Space per Interface

Similarly, bounding the label space per interface (or link) can be done by

$$\sum_{m,n\in\mathbb{N}} \left(z^m_{(i,j)} + \hat{z}^n_{(i,j)} \right) \leq L_{(i,j)}, \forall (i,j) \in E \tag{4.5}$$

with $L_{(i,j)}$ being the maximum number of labels *per link* (or interface) in (i,j).

4.5.4 No Label Stacking

For the particular case of the ROUTING AND LABEL MERGING problem, the previous model can be simplified by eliminating the decision variables $\hat{z}^n_{(i,j)}$ and $y^{\phi,\alpha}$. This leads to the removal of inequalities (4.1e) and (4.1f), and the modification of inequalities (4.1g) accordingly. This variation is mainly used for modeling ELS (see Section 4.6.3).

4.6 Simulation Results

Since the purpose of this book is to delve into the value of mathematical formulations of networking problems, in this section we give a brief explanation of the simulation results of the aforementioned problems.

Since the motivations and objectives are different for every technology, the results are depicted in different subsections following the same order as that in Section 4.2.

4.6.1 MPLS-TE

MPLS-TE was designed for the support of external path computation procedures aiming at optimizing some TE metric. Therefore, it is the aim of MPLS-TE to rely on routing protocols for path computation. Hence, for MPLS-TE, we present simulation results that do *not* include the routing part of the problem. That is, we are given a set of paths and we only want to reduce the number of labels by label merging and stacking.

We want to compare how many labels can be reduced by using the stack. An extended version of the results presented in this subsection can be found in [16].

We considered an Australian ISP topology gathered in the Rocketfuel project. The topology consists of 28 nodes. We choose 30% of nodes as ingress and egress and we vary the number of routed LSPs between them from 10 to 300. To generate

the path routes, we use the k-shortest path first (k-SPF) algorithm. We have chosen
k-SPF because shortest path routing leads to the worst link utilization. As a conse-
quence, the usage of k-SPF would give us a lower bound on the usage of the label
space incurred by any of the traditional routing solutions.

Figure 4.8 shows the number of labels used when: *a*) no reduction method is
applied, *b*) label merging and no label stacking is used and, *c*) label merging and
stacking is used. The numerical value is found using the ILP formulations described
in the chapter. At the bottom of the figure we show the average overhead in the
packets due to the usage of the stack.
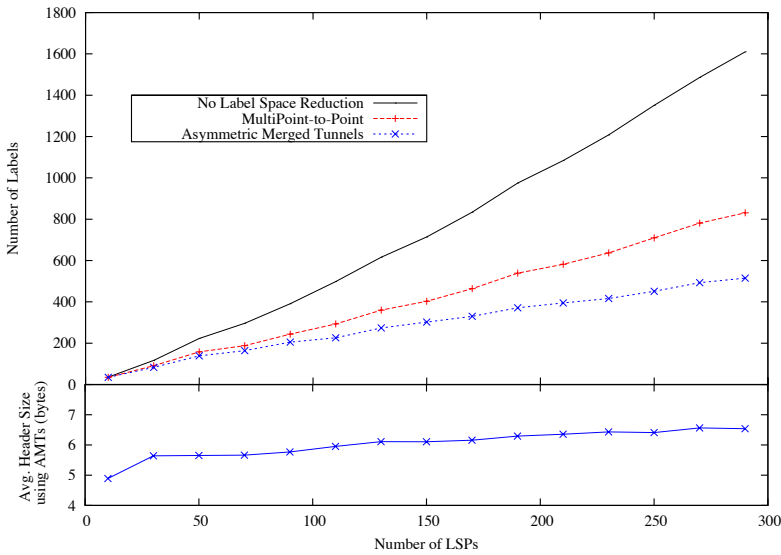
Fig. 4.8 Comparison using one stacked label. The routes are given

We observe that, when the network load is high (300 LSPs), the label space
reduction is 70.6% using the stack, while without using the stack, label merging
achieves a reduction of 48.75% in the label space usage.

4.6.2 AOLS

The motivations for reducing the label space usage in AOLS are as strong as con-
sidering the selection of appropriate routes within the same problem. That is, to
consider the path routes that would help reduce the label space, as mentioned in
Section 4.4. Since the tendency of an 'optimal' routing solution is to saturate link
capacity, we summarize in this subsection results showing the trade-off between link

capacity and the number of labels that can be saved. An extended version of these results can be found in [11].

The results we show in this section are computed using a heuristic. However, they are corroborated with the optimal solution of the model in smaller networks. The gap between the heuristic and the model solutions lies within 20% of the difference when considering smaller networks. We use two routing heuristics: Constraint Shortest Path First (CSPF) and Path-Interfering Routing Algorithm (PIRA) (specifically designed for label space usage optimization) [11].

In brief, we found that the use of the stack reduces the label space four times on average when the capacity of the links is just enough to route traffic, and almost six times if the link capacity is doubled.

Considering the routing solutions, we noticed that while the Maximum Link Utilization (MLU) of CSPF is 934 units of traffic, PIRA's is 2,236; this is 2.5 times more. However, we notice that this case occurs in few links. Figure 4.9 shows the distribution of links in PIRA that are above a given ratio of CSPF's MLU. For instance, there are six links in PIRA that are using between 50% and 75% *more* capacity than the minimum MLU considering CSPF routing. It turns out that while 25 links (out of 114) require a higher link capacity, 74 are not used by PIRA (16 of them are not used by CSPF either).

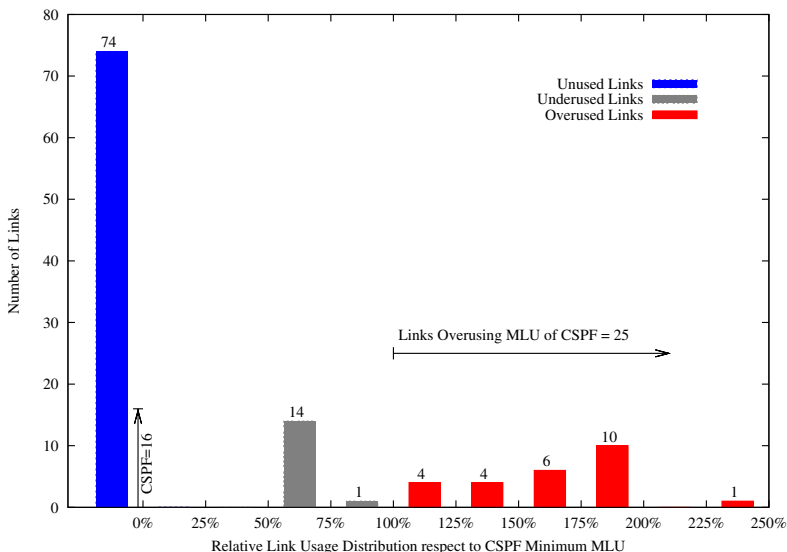**Fig. 4.9** Distribution of links in PIRA that are exceeding CSPF's MLU

We now compare the best solution without stacking (CSPF with label merging) with the best solution using the stack (PIRA with label merging and stacking). The maximum number of labels that the label merging solution uses is 20. This makes all AOLS blocks sized for coding five-bit long labels. By using stacking, the maximum

number of labels becomes 11, making the AOLS blocks sized for coding labels only four bits long. In our results without stacking, we noticed that 11 links are causing the five- bit long labels. However, using the stack, only four links are forbidding us from using three-bit long labels. Even though we did not optimize the maximum number of labels per link, it is not difficult to see that rerouting the traffic in the three links of our solution would be easier than rerouting the traffic in the 11 links in the label merging solution in order to reduce by one bit the label encoding size.

4.6.3 ELS

As mentioned in Section 4.2.3, the motivations to study ELS labels space are to determine the scalability of the architecture and the available methods when using a 12-bit label. This has been mainly studied in [3]. As in MPLS-TE, in [3], the improvement of label space usage has not been considered as a routing objective. Instead, the scalability of ELS was evaluated for the off-line and online routing scenarios. Both cases, when labels have link and node scope, were considered.

For the online routing scenario, the Shortest Path First (SPF), the Constraint Shortest Path First (CSPF), and the MIRA were implemented. Three topologies were considered, COST266, Germany50 and Exodus(US); a capacity of 10 Gbit/s are assigned to all the links [3]. Two sets of LSP requests were evaluated, one with requests of low capacity (1 Mbit/s) to consider a worst case (given that label sparcity is higher for low capacity demands) and another with requests of different capacities (1 Mbit/s, 2 Mbit/s, 10 Mbit/s, 20 Mbit/s) to consider a more realistic case. Results are compared in terms of the decrease in throughput given by the rejection of LSP requests due to unavailability of labels.

- When considering homogeneous bandwidth requests of 1 Mbit/s, results show that with the label size restricted to 12 bits per link, all evaluated algorithms result in a decrease in throughput that ranges from 32% to 50% compared to the scenario in which there is no limit in the label space size. With a restricted label size but label merging enabled, the resulting throughput is identical to the one obtained when using an unlimited label size. When the label size is restricted to 12 bits per node, the decrease in throughput ranges from 46% to 69% and with label merging enabled from 12% to 22%. The latter observation applies for all the evaluated algorithms. This is an interesting result as it shows that label merging overcomes the label size limits for a link scope even with demands of low capacity. This is not the case when the labels have a node scope were there are limitations even with merging.
- When considering heterogeneous bandwidth requests, with the label size restricted to 12 bits per link, none of the evaluated algorithms show a decrease in throughput higher than 1%. In addition, when label merging is applied, the maximum number of labels used decreases considerably (from 42% to 57%). When the label size is restricted to 12 bits per node, the decrease in throughput ranges from 19% to 29%, and with label merging enabled, from 1% to 11%.

For the off-line routing scenario model, a model similar to the one presented in Section 4.5 with the no label stacking and Traffic Engineering Variations was implemented. Results show that the highest maximum number of utilized labels was very low compared to the unused labels for both node and link scopes. This result shows that, even without merging, for the off-line scenario a 4,096 label value space is not a limitation.

4.7 Conclusions and Future Work

In this chapter we have explained and modeled the LABEL SPACE REDUCTION problem in MPLS-TE, AOLS and ELS. We reviewed two methods for reducing the label space: label merging and label stacking. A mathematical optimization model of the problem has been proposed, taking into account technical restrictions for each technology.

In brief, we noticed that the usage of a label stack, when allowed by the technology, yields to a 50% reduction in the label space usage. In scenarios in which we can set up favorable routes, simulation experiments showed that the label space can be reduced six times using the stack if the MLU limit (or link capacity) is increased at least twice. If the MLU is kept to the minimum needed by the traditional CSPF routing, the use of the stack yields up to four times label space reduction.

For the case of ELS, where the length of the label is set to 12 bits and label stacking is not supported, simulation results show that label merging overcomes the scalability limitations for the studied scenarios and link scope labels. However, complementary studies considering more topologies and different scenarios need to be done in order to fully determine the scalability of the technology.

The label reduction mechanisms proposed in this chapter are technology-specific, i.e., for each technology a specific mechanism has been proposed and evaluated. The next step would consist of unifying the techniques for multilayer networks (which implies unified label management), proposing time and resource-efficient resilience mechanisms and extending the proposed techniques for domain-wide label reuse (shared forwarding entries are nothing more than an inverse multiplexing tree). Another topic for further investigation is related to the impact on traffic flowing through label-merged data paths. Studies comprising the effect of label merging in terms of statistical multiplexing and QoS aspects would complement the derivation of the gain with respect to the forwarding plane resources.

References

1. Awduche, D., Berger, L., Gan, D. H., Li, T., Srinivasan, V., Swallow, G.: Extensions to RSVP for LSP Tunnels. IETF (2001). RFC 3209
2. Caenegem, R. V., Colle, D., Pickavet, M., Demeester, P.: Benefits of label stripping compared to label swapping from the point of node dimensioning. Photonic Network Communications Journal **12**(3), 227–244 (2006)
3. Caro, L. F., Papadimitriou, D., Marzo, J. L.: Improving label space usage for Ethernet label switched paths. In: Proc. IEEE International Conference on Communications (ICC 2008) (2008)
4. Ciavaglia, L., et al.: TIGER: Optimizing IP & Ethernet adaptation for the Metro Ethernet market. In: Proc. European Conference in Networks and Optical Communications (NOC 2007) (2007). Invited Paper.
5. Kodialam, M., Laksham, T.: Minimum interference routing with applications to MPLS traffic engineering. In: Proc. IEEE Infocom 2000, vol. 2, pp. 884–893 (2000)
6. Kompella, K., Rekhter, Y.: Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE). IETF (2005). RFC 4206
7. Mannie, E.: Generalized Multi-Protocol Label Switching (GMPLS) Architecture. IETF (2004). RFC 3945
8. Papadimitriou, D., Dotaro, E., Vigoureux, M.: Ethernet layer 2 label switched paths (LSP). In: Proc. of Next Generation Internet Networks, pp. 620–621 (2005)
9. Ramos, F., Kehayas, E., Martinez, J. M., Clavero, R., Marti, J., Stampoulidis, L., Tsiokos, D., Avramopoulos, H., Zhang, J., Holm-Nielsen, P. V., Chi, N., Jeppesen, P., Yan, N., Monroy, I. T., Koonen, A., Hill, M., Liu, Y., Dorren, H., Caenegem, R. V., Colle, D., Pickavet, M., Riposati, B.: IST-LASAGNE: Towards all-optical label swapping employing optical logic gates and optical flip-flops. IEEE J. Select. Areas Commun. **23**(10), 2993–3011 (2005)
10. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. IETF (2001). RFC 3031
11. Solano, F., Colle, R. V. C. D., Fabregat, R., Pickavet, M., Marzo, J., Demeester, P.: All-optical label stacking: Easing the trade-offs between routing and architecture cost in all-optical packet switching. In: Proc. IEEE Infocom 2008 (2008)
12. Solano, F., Fabregat, R., Donoso, Y., Marzo, J.: Asymmetric tunnels in P2MP LSPs as a label space reduction method. In: Proc. IEEE International Conference on Communications (ICC 2005), pp. 43–47 (2005)
13. Solano, F., Fabregat, R., Marzo, J.: A fast algorithm based on the MPLS label stack for the label space reduction problem. In: Proc. IEEE IP Operations and Management (IPOM 2005) (2005)
14. Solano, F., Fabregat, R., Marzo, J.: Full label space reduction in MPLS networks: Asymmetric merged tunneling. IEEE Commun. Lett. **9**(11), 1021–1023 (2005)
15. Solano, F., Fabregat, R., Marzo, J.: On optimal computation of MPLS label binding for multipoint-to-point connections. IEEE Transactions on Communications **56**(7), 1056–1059 (2008)
16. Solano, F., Stidsen, T., Fabregat, R., Marzo, J.: Label space reduction in MPLS networks: How much can a single stacked label do? IEEE/ACM Trans. Networking **16**(6), 1308–1320 (2008)

Chapter 5
Network Survivability: End-to-End Recovery Using Local Failure Information

José L. Marzo, Thomas Stidsen, Sarah Ruepp, Eusebi Calle, Janos Tapolcai, and Juan Segovia

Abstract This chapter presents an advanced shared protection approach called Failure Dependent Path Protection (FDPP). Under this approach, several protection paths can be assigned to connections in the context of a shared protection framework. After formalizing the survivable online routing problem, two possible implementations are compared, one based on heuristics and the other on ILP. Building upon the concepts of routing already exposed, the chapter then presents two case studies. The first one employs Shortcut Span Protection to examine how different protection strategies affect resource provisioning, while the second is a thorough analysis of the performance of path protection in terms of connection availability, both for dedicated and shared path protection in heterogeneous network topologies.

Key words: network survivability, protection, restoration, failure dependent protection, multi-commodity connectivity, shortcut span protection, connection availability

José L. Marzo · Eusebi Calle · Juan Segovia
Institute of Informatics and Applications, University of Girona, Campus Montilivi, 17071 Girona, Spain, e-mail: `joseluis.marzo@udg.edu`, e-mail: `eusebi.calle@udg.edu`, e-mail: `jsegovia@eia.udg.edu`

Thomas Stidsen
Informatics and Mathematical Modeling, Danish Technical University, Richard Petersens Plads, DK-2800 Kgs. Lyngby, Denmark, e-mail: `tks@imm.dtu.dk`

Sarah Ruepp
Networks Competence Area, DTU Fotonik, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark, e-mail: `sr@com.dtu.dk`

Janos Tapolcai
Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary, e-mail: `tapolcai@tmit.bme.hu`

5.1 Basic Concepts on Network Survivability

Network *survivability* reflects the ability of a network to maintain service continuity during and after failures. Although this ability is of relevance for any transport network, it is essential for operators of optical networks, where a single fiber cut, for example, can lead to severe service disruption and to loss of revenue, affecting thousands of end users whose traffic is transported by high-capacity DWDM links. Naturally, failures are not limited to fiber optic cables; other components such as multiplexers, optical cross-connects (OXC), and repeaters can also fail. Moreover, the causes of failure are equally diverse, from aging of the physical components to natural dissasters to errors caused by human intervention.

Recovery is the name given to the sequence of events and actions taken after the detection of a failure in order to keep the service in operation –whether in degraded mode or not– and return the network to the preferred state upon the completion of the repair procedures [20]. In this chapter, we consider that the unit of service of the optical network, and therefore the subject of recovery, is a *connection*, i.e., the virtual communication channel created between two designated nodes, with a given capacity and duration. In line with the concepts of GMPLS, we assume that any such connection is served by paths inside the network.

The body of knowledge on network recovery for optical networks is extensive. However, as this chapter is focused on end-to-end recovery using local failure information, the reader interested in the general topic of network recovery is referred to [13], which surveys the major techniques for next generation networks, and to [6], which offers several options for classifying them, as well as an evaluation of their features from the perspective of quality of service and differentiation. Nevertheless, we will devote the following subsections to presenting essential concepts and terminology on survivability, necessary for understanding the problems addressed in the rest of the chapter as well as the applicability of the proposed solutions.

5.1.1 Protection and Restoration

The existing recovery techniques differ in objectives (i.e., offer very fast recovery time, maximize network utilization, accomodate conflicting QoS requirements, or a combination of several objectives) as well as approaches to specific issues such as the provisioning method employed, the protection scope, the signaling requirements, and the layers, topologies and transmission technologies to which they are applicable. Despite their differences, however, recovery generally implies that the traffic affected by a failure is switched to a *backup path*. The moment in time in which this backup path is established gives rise to two general approaches, one is called *restoration* and the other is called *protection*. Under *restoration*, backup paths are discovered on demand, and spare capacity is dynamically allocated thereafter, whereas under *protection* both steps are completed at service setup time, whether any failure arises or not later on during the service lifetime.

In general, protection favors quality of recovery in the sense that, by doing pre-allocation, there is guarantee that resources are readily available at failure time and less steps are necessary to successfully complete the recovery process. But this guarantee comes at the expense of network utilization: capacity for recovery is essentially locked even if finally it is never needed. As discussed in Section 5.1.3, network utilization can be improved by sharing the resources for recovery at the cost of losing the certainty on successful recovery.

5.1.2 The Scope of Backup Paths

The conceptually simplest method of path assignment is called *global backup path* or just *path protection*, whereby two disjoint paths are assigned to one connection, as illustrated in Figure 1(a). The path that carries the traffic under normal conditions is called the *working path*, and it is covered by the backup path. When the working path is affected by a failure, the connection is rerouted on an end-to-end basis, that is, the switchover is usually performed at the source node, irrespective of the actual failure location. Unless *1+1 protection* is used (discussed in the next subsection), the source node must be notified upon failure, which increases the recovery time as well as the risk of losing traffic. Its advantage is that only the source node needs switchover functionality.
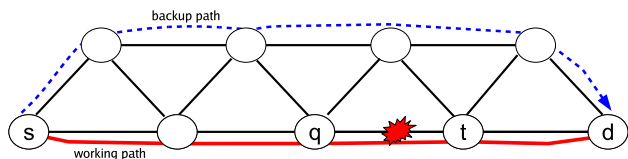
(a) Global backup path
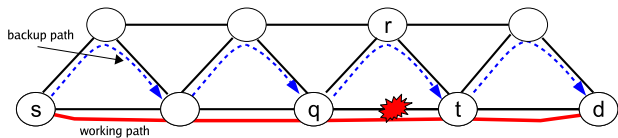
(b) Local backup path
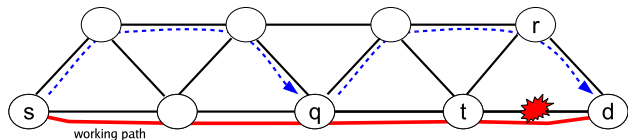
(c) Segment-based backup path

Fig. 5.1 Scope of backup paths

The *local path* method can be employed to overcome the drawbacks of using global backup paths. As illustrated in Figure 1(b), the working path is divided into shorter reroutable paths so that the node detecting the failure (node *q* in the example) can immediately switch to the alternate path (passing through *r* in the example) and route the connection around the failed element. However, this method requires switchover functionality in every node.

An intermediate solution is offered by the *segment-based* method [4]. For instance, if a failure is detected between nodes *t* and *d* in Figure 1(c), the failure notification is sent to *q*, which performs the switchover. The nodes between the failed network element and the source node of the connection are called *upstream nodes*, while the ones between the failure and the destination node are called *downstream nodes*.

The idea of restricting the notification to the node closer to the point of failure is employed by the *local-to-egress restoration* method [2]. In this case, the affected traffic is rerouted between the upstream node adjacent to the failure and the egress node of the connection, combining short notification time and high resource efficiency. This form of recovery is used by the Shortcut Span Protection method presented in Section 5.4.1.

5.1.3 Shareability of Protection Resources

Based on whether or not the sharing of network resources between connections is allowed, a protection scheme can be categorized as *shared* or *dedicated*. An example of dedicated protection that uses global backup path is *1+1 protection*. In 1+1 protection, traffic is sent simultaneously over both the working and the backup paths; the destination node monitors continuously the reception for choosing the most convenient path. By not requiring failure notification, the recovery time can be very short. Another variation is called *1:1 protection*, in which, unlike in 1+1 protection, traffic is sent over the working path only until a failure is detected.

In *shared protection*, however, the backup paths are provisioned only at the control plane to facilitate the sharing of the recovery resources of several working paths. It is also called soft provisioning [34] in GMPLS terminology and backup multiplexing [23] in optical transport domain. Shared protection is usually combined with single link failure protection. This combination yields to low bandwidth utilization and is relatively simple to implement. Section 5.4.4 studies the availability of connections protected with both dedicated path protection (DPP) and shared path protection (SPP).

The concept of shared protection is to activate a single protection path after the interruption of working bandwidth at a preselected upstream node (a branch node in GMPLS terminology [34]); the protection path merges back to the working route at a preselected downstream node called merging node. In SPP, the switching node is always the source node, while the merging node is always the destination node. Shared Link Protection (SLP) typically protects link failures only, and the switching

node is the neighboring upstream node, while the merging node is the neighboring downstream node. SLP is favored for its simple and local fault recovery. For Shared Segment Protection (SSP, a.k.a. Sub-Path Protection [25]) the working path is divided into segments and the starting node of the segment is the switching node and the last node is the merging node. The segments may be overlapped but not embedded; thus, the closest upstream switching node is always the switching node of the corresponding segment. It provides an explicit mapping between the network element and the segment. SSP provides the finest compromise between fast restoration time and bandwidth utilization efficiency [34].
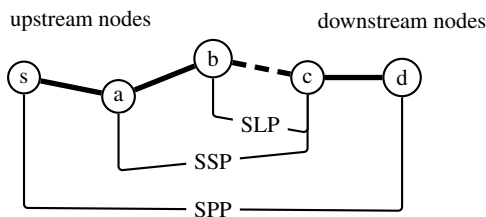
Fig. 5.2 Shared path/segment/link protection

5.2 The Failure-Dependent Path Protection Method

Shared protection is a well-studied area of survivable routing and a great number of shared protection methods have been published; however, only very few failure dependent path protection methods were studied. One of the main reasons of the scant proposals is that the problem can easily and efficiently be solved with a simple heuristic based on shortest path search. This simple heuristic is referred to as the SPH approach later in this section. In [37] the problem is called partial path protection. Besides the SPH approach, in the next section an ILP formulation is given that provides a solution with optimal bandwidth utilization in optical networks. We proceed now with detailed study of the corresponding routing problem, propose some novel approaches, and further verify the benefits of the SPH approach.

5.2.1 Recovery Based on the Failure Scenario

A common point of shared protection is that the preselected upstream node does not need to know which network element has failed. This approach is called failure independent (FI) [28] or state-independent [38] protection.[1] Note that SLP for single

---

[1] The "state" means network failure state, indicating the failed network component(s).

link failures is a special case where basically the switching node has the knowledge of the failed link; however, it is still considered as being failure independent. It is because from the operational point of view a single protection path is assigned for each switching node.

Alternatively, the connection may be assigned more than one protection path, depending on the failure scenario. Upon a failure, the switching node activates a protection path corresponding to the failed network element. Such an approach requires precise knowledge of the failure in the network; hence, it is referred to as failure-dependent (FD) [28] or state-dependent [38] protection.

Similar methods have been published related to path restoration. Contrary to all other methods discussed, path restoration is not a preplanned protection scheme; thus, the path computation takes place after the failure occurs. However, the required minimum spare capacity is basically determined with FD protection routing methods. In [7] the method is called true-path restoration, in [11, 27] simply path restoration, and in [21] and [17] path restoration with static traffic. As FD protection is tailored to specified failures, normally it requires less spare capacity than FI protection [18]. Even though FD protection was neglected due its longer restoration time and for the extended nodal processing and memory requirements [38], the Internet Engineering Task Force (IETF) solved this problem and published RFC 4090 [26], which addresses the necessary signalling extensions to support a recovery scheme called MPLS fast reroute. The IETF fast reroute defines two methods. The first one is called one-to-one backup, where each Label Switched Path (LSP) is protected separately. Among the FI methods, shared segment protection follows very similar ideas. The second method is called facility backup and each facility is protected with a single protection bypass tunnel between the potential failure points; thus, any LSP passing through the facility is protected by the same bypass tunnel. P-cycles can be treated as a similar approach for the FI case.

We assume that each working path is protected separately and we consider one-to-one backup. The key advantage of MPLS fast reroute is that it provides FD protection with short restoration time. This can be done by fixing the switching node as the first upstream adjacent node, while the merging node can be any of the downstream nodes. The failures of the network elements are detected by the adjacent nodes and, since only single link failures are considered, we may assume that the switching node has knowledge of the failed link after the failure detection time, leading to a rapid recovery cycle. In our study we allow any upstream node to be a switching node; however, the above limitation can simply be applied to the proposed algorithms. In [36] the MPLS fast reroute is extended with distributed shared bandwidth management, which allows sharing of recovery resources of disjoint working paths.

5.2.2 Path Assignment Approaches

We consider an online routing problem, without any knowledge of future request arrivals and without applying prediction-based routing on the statistics of past requests. Traffic Engineering (TE) controls traffic to assure an economical utilization of network resources. We use link weight setting methods, which identify the bottleneck links in the network and, by assigning high administrative link weights, circumvent these links during the course of path selection. This ensures that residual capacity is always available at bottleneck links, thus facilitating the successful routing of as many future requests as possible. The path selection schemes use the administrative weights as their cost functions to take network-wide TE policies into account to achieve global optimization of network resources.

We assume source routing with complete routing information scenario, where the link state protocols disseminate all the necessity routing information to each node, including the free and spare capacities on links, the shareability of protection routes, and the administrative link weights.

5.2.3 General Shared Risk Groups (SRG)

A *Shared Risk Group* (SRG) is defined as a group of network elements (links, nodes, physical devices, software or protocol identities, etc., or a mix of them) possibly subject to a common risk of single failure. In practical cases, an SRG may contain several seemingly unrelated and arbitrarily selected network elements. We say that a working path is *involved* in an SRG if it traverses any network element that belongs to the SRG.

Most of the past studies focused on the case where each single network element in the network topology serves as an SRG. Even if this special case is widely accepted and very common, we believe that a sterling survivable routing algorithm should be able to cope with the general definition of SRG.

Obviously, in single-layer and single-domain networks, multiple network elements might be contained in a single SRG. However, the concepts of general SRG particularly contributes to the development and implementation of survivable routing schemes for the modern multilayer, multi-domain, and multi-carrier public networks. Because the network can be multilayered, it is not straightforward to take the network elements in the upper virtual layer and the underlying physical layer in a common SRG, although the upper layer virtual topologies could be embedded in the lower-layer topology. The general SRG concept simplifies this dependency by separately grouping the network elements of each layer. In the case of a multi-domain and multi-carrier network environment, a suite of efficient and secure link state information dissemination mechanisms must be developed to support routing in the network layer. Here, the definition of general SRGs can help the link state information aggregation, classification, and encapsulation to achieve a resource-sharable protection plan.

In end-to-end FI protection (e.g., dedicated or shared path protection), the task is to find an SRG-disjoint working and protection path-pair for a connection request, which has been proved to be NP-complete [1, 9, 15]. Therefore, the problem is either solved by heuristics [11, 14] or by exponential worst-case algorithms, like Integer Linear Program (ILP) [14, 15].

The consideration of the general definition of SRGs has two impacts upon the solving of the survivable routing problem compared to the case where there is a one-to-one mapping between a link or node and an SRG. Firstly, solving the survivable routing problem turns out to be solving an SRG-disjoint path-pair, which is NP-hard. In addition to the NP-completeness, the consideration of the general definition of SRGs may introduce an increase in the size of the spare provision matrix (SPM) [22]. In [33] the matrix expression by Yu Liu [22] was modified to enumerate the Spare Provision Matrix in the case of general SRG.

5.2.4 The Input of the Problem

Given a network with a set of nodes N and a set of links L, a corresponding transformed graph can be produced by modeling each network element of interest in the original network as an arc in the transformed graph. Each SRG of the original network can be represented by a set of arcs in the transformed graph.

Let $G(V, E)$ denote the transformed graph of the original network with a set of *arcs E* and *vertices V*, where $|E|$ and $|V|$ are the number of arcs and vertices in G. The cost for allocating a unit capacity on arc j (the administrative weight) is denoted as $c_j \ \forall j \in E$. The unreserved free capacity along arc j is denoted as $f_j \ \forall j \in E$. The amount of capacity reserved along arc j is denoted as $v_j \ \forall j \in E$. Furthermore, we are given the source vertex s and the destination vertex d of the new demand with a specific amount of bandwidth b. Due to the complete routing information scheme, the full per-flow information of the network (i.e., the working and protection paths along each link) is known. Based on the full per-flow information, the Spare Provision Matrix (SPM) can be calculated [33]. It is denoted as $\underline{\underline{S}}$ and a $|E| \times |SRG|$ matrix. The entry (i, j) of $\underline{\underline{S}}$ (denoted as $s_{i,j}$, where $i = 1 \ldots |E|$), $j = 1 \ldots |SRG|$, is the amount of non-sharable spare capacity along arc i for P if W is involved in the jth SRG.

The feasible condition of the primary (a.k.a working) path is $f_j \geq b$ for all arcs $j \in W$ (the working path). In FDPP a backup path is assigned to each SRG involved in W. The feasible condition of the backup path P_j assigned to the jth SRG involved in W is $f_i + v_i - s_{i,j} \geq b$ for for all arcs $i \in P$.

5.2.5 Two-Step Approaches

In two-step approaches the optimization is divided into two steps. First, a shortest path is found and assigned as the working path; second, the protection paths are identified.

Two-step approaches are widely used in shared path protection due to their simplicity and decent performance, even if they cannot cope with the trap problem [39]. In the trap problem, the shortest path is such an unfortunate working route that it has no SRG-disjoint protection path, even if there exists an SRG-disjoint path-pair between the given source-destination pair. Due to the trap problem, the two-step approaches have always higher blocking than approaches in which the working and protection paths are jointly optimized.

There are two main concepts that can be applied to solve the trap problem in failure-independent protection environments. First is the selection of a more appropriate path than the shortest one as working route. Usually, the shortest path that has a disjoint counterpart [32, 39] is selected as the working path. In the case of online routing, this can be done only with heuristics approaches, since the problem is NP-hard [19]. The second concept is to apply a protection method other than end-to-end protection. Segment protection is a good choice, since it is impervious to the trap problem. In [12] it was stated that *"In any network topology, whenever two disjoint paths exist between a pair of end nodes, backup segments are guaranteed to exist for any choice of a primary path between them. Similar guarantees cannot be provided on the existence of end-to-end backup."*

Similarly to segment protection, the trap problem can be easily handled in FDPP. Basically, without knowing the exact route of the working path, we are able to decide whether or not a link belonging to it can be protected. This can be done by simulating the failure of each SRG involved and searching for a feasible protection path between the source and destination nodes of the request. If there is a feasible protection path P after the failure of SRG a ($\forall e \in P \quad f_e + v_e - s_{e,a} \geq b$), we can be sure that P will intersect the working route in an upstream node (in the worst case at the source node), which can be treated as the switching node, and P will also intersect the working path in a downstream node (in the worst case in the destination node), which can be treated as the merging node, and thus we can be sure that there will be a feasible protection path. Obviously if there is no protection path that can protect the failure of SRG a, we will not be able to protect the working path passing through SRG a.

In the same way we can define an FDPP test that filters out all the infeasible edges from being part of the working route, and leaves all the edges that can be freely selected to guarantee a feasible FDPP protection solution.

Definition 5.1. FDPP test of arc a is true if there is a path P between s and d such that $f_e + v_e - s_{e,a} \geq b$ for all $e \in P$.

5.2.5.1 First Step

We propose two techniques for the first step of the two-step approach. The traditional way is to assign the shortest path to the working route in the graph composed of all the links with sufficient free capacity ($f_e \geq b$), with the administrative costs assigned to the links. Since the working path cannot be shared with other requests, it is a natural to attempt to dedicate the fewest possible resources for the working path.

In the second technique, we solve the trap problem with the FDPP test, and first we identify the links that can be part of the working path to get a feasible solution. After erasing all the unusable links along with the links with insufficient free capacity ($f_e > b$), we take the shortest path in the residual graph as the working path, with the administrative costs assigned to the links.

5.2.5.2 Second Step

The task in the second step is to find protection routes that require the allocation of minimal spare capacity weighted by the administrative link costs. The two-step approach is preferred for FI shared path protection due to its simplicity. In the second step of shared path protection an optimal protection path is derived with respect to a working path, which can be simply done by constructing a residual graph [32] composed of all the links with sufficient free and sharable capacity for protecting the working path. In this residual graph, the shortest path between the source and destination nodes with link costs updated according to the resource usage status is the optimal protection path. The administrative link cost is proportionally assigned to the links according to the fraction of capacity that cannot be shared by other protection routes: the full administrative link cost is assigned if none of the capacity can be shared by other protection paths, while only a very small positive cost represented by ε is assigned if it can be fully shared [32].

The problem we face in the second step of FDPP of the two-step approach is not that simple to solve. Conversely, the trap problem is a hard problem for shared protection, whereas it can be easily handled for FDPP.

Instead of solving the optimal protection path problem for a given working route in the case of FDPP, we define a generalization of the problem, which we call the Multi-commodity Connectivity (MCC) problem. In Section 5.3 we propose several methods for solving the MCC problem that can be applied in the second step of the FDPP two-step approach.

5.2.6 Joint Optimization: The Greedy Approach

In joint optimization, we consider the problem of minimizing the use of new resources (for both the working and the protection paths) required to accommodate

the new call without disturbing the existing ones. We call this approach the greedy approach. In Section 5.3.3 we formulate the traditional FDPP with ILP. This formulation seeks, upon call arrival, to minimize the total amount of resources used for establishing both primary and backup paths.

5.3 Multi-commodity Connectivity (MCC)

The *input* of the MCC problem is

- a graph $G(V,E)$ with a set of *edges E* and *vertices V*
- a set of commodities C. The number of commodities is denoted by $k = |C|$. For each commodity $a \in C$, the following input is defined:

  - a source node $s_a \in V$ and a destination node $d_a \in V$
  - a subgraph (V, E_a) with usable edges
  - a cost $c_{e,a}$ for each edge $e \in E_a$

The objective is to find for each commodity $a \in C$ a single path, denoted by P_a, with minimum overall cost, such that P_a connects s_a and d_a on edges of E_a. The overall cost of edge $e \in E$, denoted by z_e, is the maximum of the edge costs assigned to the commodities passing through it, i.e., $\max_{a \in C|e \in P_a} c_{e,a}$. The objective function is to minimize $\sum_{e \in E} \max_{a \in C|e \in P_a} c_{e,a}$.

This problem is similar to the multi-commodity flow problem. In this case, however, the commodities can share the cost when using the same edges.

FDPP is a special case of MCC, where the commodities are assigned to those SRGs that are involved in the working path, but the SRG does not form a cut between s and d, so it can be protected. The source-destination of each commodity is the source and destination of the connection ($s_a = s$ and $d_a = d$ for all $a \in C$).[2] An SRG is assigned to each commodity, and the subgraph E_a is populated as follows:

- the edges involved in the corresponding SRG are erased from E_a
- the rest of the edges involved in the working path are added with cost equal to 0 ($c_{e,a} = 0$)
- the rest of the edges with sufficient capacity to protect the corresponding SRG added ($v_e - s_{e,a} + b \le f_e$) with cost of $c_e \cdot (max\{v_e - s_{e,a} + b, \varepsilon\})$, where ε is a very small positive number.

Since we are protecting single SRG failures, the spare capacity along the protection routes of FDPP can be shared, and the objective of the MCC is equal to the amount of spare capacity required to be reserved weighted by the administrative link cost.

---

[2] In the case of MPLS fast reroute, the source is required to be the adjacent upstream node of the working path.

5.3.1 Complexity of the Multi-commodity Connectivity Problem

Theorem 5.1. *Finding the minimum cost MCC solution is* NP-*hard.*

Proof. Obviously, this problem belongs to the class NP. We shall reduce 3SAT to our problem as follows: First we construct an undirected graph $G = (V, E)$. For each variable x_i $(i = 1...n)$, let p_i be the number of occurrences of the literal x_i in the clauses and p'_i the number of occcurrences of the literal $\bar{x}_i$, and construct a lobe as shown in Figure 5.3.
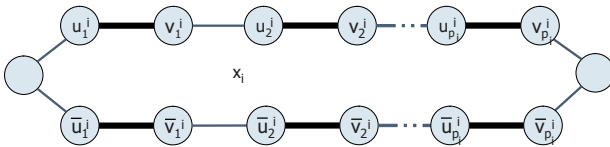

Fig. 5.3 Lobe

The lobes are connected one after the other. For each clause C_j, we add two vertices y_j, z_j $(j = 1...m)$ with edges between $(u, y_1), (z_j, y_{j+1})$ $(j = 1...m)$, and (z_m, x). Finally, we connect clauses to variables by adding the following edges:

$(y_j, \bar{u}^i_k)$ and $(z_j, \bar{v}^i_k)$ if the kth occurrence of variable x_i is the literal x_i, that is, a literal in clause C_j

(y_j, u^i_k) and (z_j, v^i_k) if the kth occurrence of variable x_i is the literal $\bar{x}_i$, that is, a literal in clause C_j

For example, the graph G corresponding to the instance

$$(\bar{x}_1 \vee x_2 \vee \bar{x}_3)(x_1 \vee \bar{x}_2 \vee x_3)(x_1 \vee x_2 \vee \bar{x}_3)$$

is depicted in Figure 5.4. E_1 consists of the edges shown as solid lines, while E_2 contains the ones drawn with broken lines plus the thick solid lines.

Edges have cost 0, except the ones shown with thick lines. For those edges, the cost is $\frac{1}{p_i}$ (or $\frac{1}{p'_i}$ respectively) for the first commodity, and $0 < \varepsilon << 1$ for the second commodity. Note that the cost of passing each lobe is 1 for the first commodity.

It is easy to see that G can be constructed in polynomial time. Therefore, it is sufficient to show that there exists a truth assignment that simultaneously satisfies all m clauses if and only if the cost

$$\min \left\{ \sum_{e \in E} \max_{a \in C | e \in P_a} c_{e,a} \right\} \leq n$$

If there exists a truth assignment τ that simultaneously satisfies all m clauses, then P_1 path passes through the upper portion of the lobe if $\tau(x_i) = false$ and passes through the lower portion if $\tau(x_i) = true$ and the cost is n. Each satisfied clause C_j contains either a literal x_i such that $\tau(x_i) = true$ or literal $\bar{x}_i$ such that $\tau(x_i) = false$,
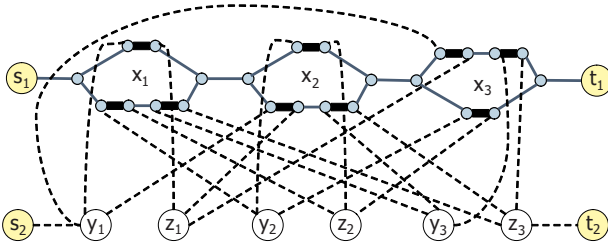
Fig. 5.4 Graph G corresponding to $(\bar{x}_1 \vee x_2 \vee \bar{x}_3)(x_1 \vee \bar{x}_2 \vee x_3)(x_1 \vee x_2 \vee \bar{x}_3)$

which implies that there exists a subpath (y_j, u_k^i, v_k^j, z_j) or $(y_j, \bar{u}_k^i, \bar{v}_k^j, z_j)$ which can fully share the cost with P_1 and thus its cost is 0.

Hence, there exist at least m such subpaths, which, together with edges (s_2, y_1) and (z_m, t_2), form a path P_2 with cost 0. Therefore, the total cost would be n.

Conversely, suppose there exist two paths with total cost m. Path P_1 must contain edges drawn with only continuous lines, and since passing through each lobe costs 1, the cost of the path would be m. We set $\tau(x_i) = true$ if P_1 passes through the lower portion of the ith lobe and $\tau(x_i) = false$ otherwise. On the other hand, path P_2 must pass through all vertices y_j and z_j, and it should share the cost with edges of P_1 in order to reach t_2 without requiring any additional cost. According to this truth assignment, clause C_j is satisfied since there is an edge in the lobe between all y_j and z_j in P_2. Therefore, all m clauses are satisfied.

A special case of this problem is called directed Steiner Network Problem [10] where, given a single directed graph G and source-destination node pairs (s_a and d_a), for each commodity the task is to find the smallest subgraph H of G that contains a path from s_a to d_a for all $a \in C$. MCC with $E_a \equiv E$ and $c_{e,a} = 1$ for all $e \in E$ and $a \in C$ represents the same problem. The latter problem in NP-hard for a general number of commodities, since the Directed Steiner Tree problem is a special case. However, the problem was proved to be polynomially solvable if k is any constant number.

5.3.2 The SPH Approach

The SPH approach is a simple heuristic where an ordering of the commodities is defined and each P_a is calculated with a shortest path search, one after the other, such that at each step the minimum additional overall cost is added. In the implementation $z_e = 0$ at the beginning and it is updated after a new path is calculated ($z_e = \max_{\forall a | e \in P_a} \{c_{e,a}\}$). Each path is calculated with a shortest path search in the corresponding subgraph (for commodity a it would be (V, E_a)), where the link costs are $\max\{0, c_{e,a} - z_e\}$.

When the SPH approach is applied for FDPP, first a working path is derived in the first step (with a shortest path search). Then a residual graph is constructed (and can be protected) for each SRG involved in the working path. In the residual graph the links involved in the corresponding SRG are erased and all the links in the working path disjoint from the SRG are available at 0 cost. Besides, it consists of all links where $v_e - s_{e,a} + b < f_e$ holds. These links have costs according to the resource usage status. In the next step a backup path is calculated (with a shortest path search) with the objective of using the least resources on the corresponding residual graph. Basically, the administrative link costs are proportionally assigned according to the fraction of the spare capacity that needs to be additionally reserved along the links: the full administrative link cost is assigned if none of the capacity can be shared by other protection paths. However, if it can be fully shared, only a very small positive cost represented by ε is assigned.

Note that the order in which the paths are established will affect the final solution. In our implementation we visit the SRGs of the working path in order of traversal from the source to the destination.

5.3.3 ILP of the Multi-commodity Connectivity Problem

Since the problem is NP-hard, we have formulated it with ILP mainly to understand the character of the problem and facilitate the introduction of some other effective heuristics. The following symbols are adopted to develop the ILP formulation. Let $y_{e,a}$ be a binary variable assigned for all commodities $a \in C$ and edges of $e \in E_a$. Variable $y_{e,a}$ represents the route of commodity a, such that it is 1 if the corresponding path passes through the arc e, and 0 otherwise. Let z_e be a nonnegative real variable assigned to the overall cost of edge e.

The MCC Problem reads

$$\min \sum_{a \in E} c_e z_e \tag{5.1a}$$

$$\text{s.t.} \sum_{(i,j) \in E_a} y_{(i,j),a} - \sum_{(j,i) \in E_a} y_{(j,i),a} = \begin{cases} 1 \text{ if } i = s_a \\ -1 \text{ if } i = d_a \\ 0 \text{ otherwise} \end{cases} \quad \forall i \in V, \forall a \in C \tag{5.1b}$$

$$c_{e,a} y_{e,a} \leq z_e \qquad \qquad \forall e \in E_a, \forall a \in C \tag{5.1c}$$

where equations (5.1b) are known as the flow conservation constraints and inequalities (5.1c) as the maximum cost constraints.

5.3.4 Examples

We briefly compare the two main implementation approaches: the joint optimization (greedy ILP) approach and the two-step (SPH) approach. Solving an ILP is computationally intensive. In contrast, since the algorithms for seeking the shortest paths, e.g., Dijkstra's algorithm, are polynomial-time, the shortest primary path approach can place a new call rapidly.

Let us now consider resource efficiency. While the SPH approach may at times require more resources for a given call, it is possible that over a number of calls, the SPH approach may eventually result in more efficient bandwidth utilization. Example 5.1 illustrates this phenomenon.

Example 5.1. Consider the network in Figure 5.5 and assume that FDPP is employed. The network is initially empty with a uniform administrative cost function (all edge costs are 1) and serves three call requests, $(1,4)$, $(6,3)$, and $(3,5)$ in sequence. Table 5.1 shows the resource assignments for the greedy approach and the SPH approach. In this example, the SPH approach initially occupies more wavelengths to support the request $(1,4)$ than does the greedy approach. However, as the calls accumulate, the SPH approach uses fewer wavelengths to support the same requests than the greedy one.
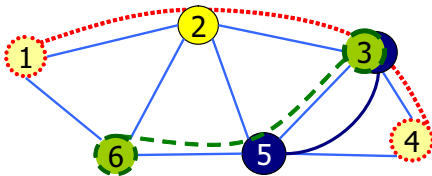
Fig. 5.5 An example network

In this example, the greedy approach endeavors to serve each request using the minimum number of previously unused wavelengths. However, in doing so, the greedy approach happens to choose paths with no protection sharing, harming network resource utilization. In contrast, though the SPH is not optimal at first, it performs better over the call arrivals by encouraging protection sharing.

5.4 Case Studies

The objective of this section is to present two case studies that employ and enhance some of the previously exposed concepts. The first one analyzes how different implementations of failure-dependent protection strategies affect the network resources that must be provisioned, and presents a new model to overcome some

Table 5.1 Resource usage for network employing the failure-dependent path protection scheme implemented by different approaches in Figure 5.5

	Primary path	Protection path (protected link)	Number of taken unit of capacity
Greedy approach	1-2-3-4	1-6-5-4 (1-2-3-4)	6 (no sharing)
	6-5-3	6-2-3 (6-5-3)	10 (no sharing)
	3-5	3-2-5 (3-5)	13 (no sharing)
Shortest path approach	1-2-3-4	1-6-2-3-4 (1-2)	7 (share (2-3-4))
		1-2-5-4 (2-3)	(share (1,2))
		1-2-5-4 (3-4)	
	6-5-3	6-2-3 (6-5)	10 (share (6,2))
		6-2-3 (5-3)	
	3-5	3-2-5 (3-5)	12 (share (2,5))

of the highlighted drawbacks. The second case study is a thorough analysis of the availability measures for connections equipped with shared and dedicated path protection in heterogeneous network topologies. It highlights the limitations of path protection to achieve very high availability even with the most resource-intensive method.

5.4.1 First Case Study: Shortcut Span Protection

This section presents a case study of how different protection strategies (such as global and local-to-egress, described in Section 5.1.2) affect the total capacity that must be provisioned in a network. With the prices for fiber (i.e., capacity) dropping [30], capacity usage will become a less important factor for deciding which protection method should be employed, whereas complexity and speed combined with the manageability of the method are expected to be given higher priority in the decision process.

   To reduce the complexity of the protection method and the restoration time, we want to investigate a variation of the local-to-egress protection method, which we have termed Shortcut Span Protection (SCSP). In SCSP, the traffic is routed from the node before the failed link directly to the endpoint, i.e., the traffic makes a shortcut. The idea is illustrated in Figure 5.6. The SCSP method achieves the same quick recovery time as standard span-protection, but is more relaxed with reference to the protection routing and hence more efficient with reference to the needed protection capacity. This improved efficiency, though, comes at the price of a more complex protection routing problem. We compare the efficiency of SCSP with that of span protection.
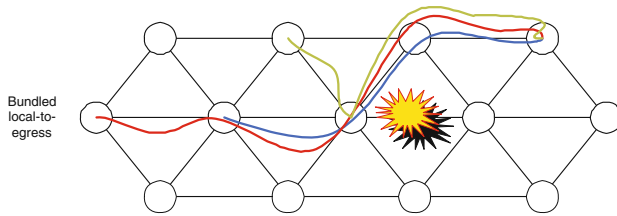
Fig. 5.6 Shortcut Span Protection (SCSP)

5.4.2 The Shortcut Span Protection Model

The routing in the SCSP model quickly becomes complicated, so we are forced to make a couple of simplifications:

- We will only consider single link failures
- We will only consider the relaxed case, i.e., the routing of the paths may be bifurcated, with both the ordinary flow and the protection flow.
- We will assume 100% protection, i.e., all traffic will be protected against any single edge failure.

To perform the routing of the nominal flow and the backup flow, we apply Linear Programming (LP). First we describe an LP model for SCSP and standard span protection. Then we present the result of the different protecion methods and comment on them.

Given a network with N nodes, we index them with several different indexes: $i, j, k, l, q, r \in N$. The indexes are used for nodes in different contexts: i, j are used to index the flow, k, l are used to index the nodes of the demand requests, and q, r are used to index the single link errors. Between the nodes there are bi-directional edges E. We will index a specific edge by its end nodes: $\{ij\}$ for the non-failed edges and $\{qr\}$ for the failed edges. All the flow is directed, and we hence represent each edge $\{ij\}$ with two arcs (ij) and (ji). When an edge fails, both arcs (qr) and (rq) fail and the flow on these arcs will have to be rerouted. We assume that a number of communication demands are given, and the volume of the demand is $D^{(kl)}$. For each demand (kl) we represent the nominal flow with a variable $x_{(ij)}^{(kl)} \in R^+$, which corresponds to the flow from node i to node j of the demand request from node k to node l. Whenever an edge $\{qr\}$ fails, both the flows (qr) and (rq) needs to be restored by a protection flow, starting from node q and node r respectively. The protection flow is represented by $y_{(ij)}^{l,(qr)}$, for the flow from node i to node j of the failed arc (qr) which is destined for node l. To ease the formulation of the model, we define an auxilliary variable $u^{l,(qr)}$ which represent the total flow with end destination node l from the failed arc (qr). We are now ready to present the complete LP model.

Shortcut span protection arc-flow LP model:

$$\min \sum_{\{ij\}} c_{\{ij\}} z_{\{ij\}} \tag{5.2a}$$

$$\text{s.t.} \sum_j x^{(kl)}_{(ij)} - \sum_j x^{(kl)}_{(ji)} = \begin{cases} D^{(kl)} & i=k \\ -D^{(kl)} & i=l \\ 0 & i \neq k,l \end{cases} \quad \forall i,(kl): D^{kl} > 0 \tag{5.2b}$$

$$\sum_k x^{(kl)}_{(qr)} = u^{l,(qr)} \quad \forall l,(qr): \sum_k D^{kl} > 0 \tag{5.2c}$$

$$\sum_j y^{l,(qr)}_{(ij)} - \sum_j y^{l,(qr)}_{(ji)} = \begin{cases} u^{l,(qr)} & i=k \\ -u^{l,(qr)} & i=l \\ 0 & i \neq k,l \end{cases} \quad \forall l,(qr),i: \sum_k D^{kl} > 0 \tag{5.2d}$$

$$\sum_{(kl)} (x^{(kl)}_{(ij)} + x^{(kl)}_{(ji)})$$
$$+ \sum_l (y^{l,(qr)}_{(ij)} + y^{l,(qr)}_{(ji)})$$
$$+ \sum_l (y^{l,(rq)}_{(ij)} + y^{l,(rq)}_{(ji)}) \leq z_{\{ij\}} \quad \forall\{ij\},\{qr\} \tag{5.2e}$$

$$x^{(kl)}_{(ij)}, u^{l,(qr)}, y^{l,(qr)}_{(ij)}, z_{\{ij\}} \in \mathbb{R}_+ \tag{5.2f}$$

The SCSP LP model consists of an objective function (5.2a), nominal flow constraints (5.2b), the constraints (5.2c) setting the auxilliary variables, the protection flow constraints (5.2d) and capacity constraints (5.2e), and domain definitions (5.2f). The objective function (5.2a) measure the cost of the necessary capacity in the network. The nominal flow constraints (5.2b) ensure that the nominal flow is routed from the start node k to the termination node l. The constraint setting the auxilliary varibles (5.2c) simply sums the nominal flow across the failed arc (qr) which shares the same termination node l. The protection flow constraints (5.2d) then use the auxilliary variables to create a protection flow. Notice that if the identification of the termination node in constraint (5.2d) is changed from l to q, then standard span protection is performed.

5.4.3 Results

In order to evaluate the effectiveness of short cut protection, we test the method, by optimizing over a set of four networks for a demand of volume 1 between all pairs of nodes in the network, that is, only one way for each pair: $D^{(qr)} = 1$ and $D^{(qr)} = 0$. Furthermore, we have set all the costs to unit values: $c_{\{ij\}} = 1$. In Table 5.2 we summarize data about the networks.

Table 5.2 Network data

Network Name	Number of Nodes	Number of edges	Average node degree
Cost239 [3]	11	26	4.73
PanEuropean	13	21	3.23
USANetwork [8]	28	45	3.21
Italy [11]	33	68	4.12

For the test networks in Table 5.2 we used the LP formulation given by the model (5.2a)–(5.2f) to test four variants of the protection methods:

1. **Shortest SCSP:** Route nominal flows on shortest paths and then protect the flows with SCSP.
2. **Shortest Span:** Route nominal flows on shortest paths and then protect the flows with span protection.
3. **SCSP:** Perform combined routing of nominal and protection flow using SCSP.
4. **Span:** Perform combined routing of nominal and protection flow using Span protection.

For each network, the required non-failure (NF) network cost is given. Furthermore, the protection lower bound on the network cost is given, based on Complete Rerouting (CR) protection [31]. For CR and the four above-described combinations of SCSP and Span protection, we give both the absolute value of the network cost and the relative increase compared to the non-failure network cost; see Table 5.3

Table 5.3 Performance of CR, SCP, and SP on four test networks

Network	NF	CR		SCSP shortest		Span shortest		SCSP		Span	
		Rel	Abs	Rel	Abs	Rel	Abs	Rel	Abs	Rel	Abs
Cost239	86	13.4	97.6	32.5	114	39.5	120	25.3	107.7	26.8	109.0
PanEuropean	158	56.9	248	70.8	270	73.4	274	66.4	263	69.6	268
USANetwork	1273	50.3	1914.2	66.5	2120.3	71.5	2184.3	60.4	2042.5	65.9	2112.0
Italy	1718	33.8	2299.3	56.8	2693.8	62.11	2785.0	46.2	2512.9	49.9	2575.6
		38.6		56.7		61.6		49.6		53.0	

From the results it can be seen that the effect of SCSP is rather small; it is only an improvement of 4% to 5% of the network cost. It turns out to be more important to perform joint routing of the nominal flow and the protection. By combined nominal flow planning and SCSP, the method on average only requires approximately 11% exstra network cost compared to the CR lower bound.

It is somewhat surprising that the benefit of SCSP is not larger. Part of the the reason for the difference not being larger is probably that stub-release is not included. Stub-release means freeing the part of the nominal flow that is not being used after a failure. This *can* be included in the LP model above, but the number of variables required grows significantly, so that only the small networks can be solved, and we have hence chosen not to include this approach here.

5.4.4 Second Case Study: Connection Availability Under Path Protection

The probability that a system (in this case a connection) will be found in the operating state at a random time in the future is called *availability*. If recovery techniques are applied, then connection availability is an estimation of the ability of the network to maintain the connection during and after a failure, i.e., it is an estimation of survivability.

As connections are served by the physical network elements assigned to them, the intrinsic reliability of these physical elements affects connection availability. A well-known availability formula for a single element (or item), employed in this study, is $A = (MTBF - MTTR)/MTBF$, where $MTBF$ is the Mean Time Between Failures and $MTTR$ is the Mean Time To Repair. These reliability measures that can be obtained from the manufacturer, be based on knowledgeable opinion, and so on.

The objective of the study presented in [29] is to analyze how connection availability is affected by different combinations of path protection schemes (dedicated vs. shared) and specific topology properties such as nodal degree, link length, and network diameter. In the following we hightlight the main results.

5.4.4.1 Context of the Study

In order to obtain heterogeneity from the point of view of topology properties, six topologies were selected with differing average node degree, number of nodes and links, network diameter, geographical coverage, and diversity in link lengths. Connection availability was evaluated with an event-based simulator considering dynamic traffic, i.e., the capacity requested by a connection was allocated and released at that connection's set up and tear-down, both events following a Poisson process with exponentially distributed connection holding times. The traffic matrix employed is from [24]. The demanded capacity was chosen randomly following a uniform distribution. In such a dynamic environment, the effectively allocated capacity depends on the protection scheme applied as well as on the routing algorithm. Therefore, an important figure of merit is *restoration overbuild*, that is, the extra capacity required for a given level of protection.

The simulation was carried out assuming no protection, shared path protection, and dedicated protection for the six topologies. The results reported correspond to an average of ten runs, and every run processed 80,000 connections.

5.4.4.2 Network Availability Model

The network availability model employed is an adaptation of the one presented in [35], developed in the context of the IST project NOBEL. Its elements are components of the physical layer (OXCs, transponders, fiber optic cable, amplifiers, etc.).

The estimation of connection availability is performed assimilating a connection to a series-parallel system. The building block in this system is the availability of each component, computed based on its intrinsic reliability measures, i.e., its MTBF and MTTR. The study was carried out using the conservative values of MTBF and MTTR proposed in [35], which are based on the opinion of the consortium's partners on the reliability of the network components included in the model.

5.4.4.3 Connection Availability

The study analyzes the availability of connections protected with DPP (1+1 protection) and SPP, as defined in Section 5.1.3. The *availability* of a connection protected with one backup path is, in general, $A = A_w + (1 - A_w)A_b$, where A_w and A_b are the availability of the primary and backup paths respectively. This formulation is directly usable with DPP, but if SPP is employed, the negative effect of sharing on the availability must be considered: if two or more simultaneous or near-simultaneous failures affect unrelated (disjoint) working paths, there is no guarantee that all of them will be recovered because they might be sharing backup resources. Specifically, a successful corrective action for one of the failing connections could leave others without the resources they need for their own recovery if the capacity for protection is already used up. A penalty for this potential access conflict should then be considered when using SPP, lowering the availability initially estimated. As explained in [16], under the assumption of independent failure events and a binomially distributed failure number in a backup path sharing group, the aforementioned penalty can be approximated with (5.3)

$$P = (1 - A_w)A_b(1 - A_w^{n-1})\left[1 - \frac{1}{2 + (n-2)(1-A_w)}\right], \tag{5.3}$$

where n is the size of the backup path sharing group.

With respect to *restoration overbuild*, further attention is also needed when SPP is applied, as the capacity effectively allocated for backup has to be accounted for. Equation (5.4), suggested in [5], was used to compute the backup capacity allocated to a given connection i:

$$C_i = \sum_{m=1}^{k} CT_m \left(\frac{d}{\sum_{j=1}^{p} C_{m,j}}\right) \tag{5.4}$$

where k is the number of links in the backup path of connection i; p is the number of backup paths that exist when this new connection is being set up; CT_m is the capacity effectively allocated for sharing purposes in link m, based on the demands of all the connections whose protection paths traverse link m; d is the demand of connection i; and $C_{m,j}$ is the capacity required by all the backup paths that are sharing link m, irrespective of the capacity effectively allocated.

During the simulation, after determining the paths assigned to an accepted connection, its availability was computed taking into account the specifics of the protection scheme. The values reported are averaged over all accepted connections.

5.4.4.4 Routing Strategy

Routing was carried out along the shortest path, the metric being distance (in km). Candidate shortest paths that did not have enough capacity for the arriving demand were filtered out. Node-disjoint working and protection paths were chosen. Blocking resulted when no path could be found to satisfy a demand with a given protection scheme. Dijkstra's algorithm was used to find shortest paths, and a two-step approach was applied to select protection paths.

5.4.4.5 Results

Table 5.4 presents the average *connection availability* under both DPP and SPP. In the case of continental-size network topologies (Cost266, KL, Janos-US-CA, and NSFnet), the average availability values are very different compared to the small or medium size networks, where no differences can be observed, independently of the application of DPP or SPP. In the case of SPP, however, the worst-performing one (Janos-US-CA) has more than six times less availability than the best (KL) one. Nonetheless, these larger topologies can only achieve a maximum of "three nines" of availability, even when DPP is applied. On the other hand, results show that the dominant component in the network availability model is the fiber optic cable, due to the frequency of cable cuts and the relatively long duration of repair times. Therefore, connection availability is dependent on the lengths of links, or in general terms, on the link length distribution of its network topology. Figure 5.7 shows how the mean downtime of connections in two of the studied topologies (Germany50 and COST266) increases when the link lengths are multiplied by a given factor; Figure 7(a) shows that the average downtime increases linearly when connections are unprotected, with different slopes for different topologies. When protection is applied, however, the degradation rate is lower (see Figure 7(b)). With respect to the *restoration overbuild*, it can be seen in Table 5.4 that DPP requires almost two times the working capacity for protection, while DPP requires approximately the same capacity for both working and protection paths.

Table 5.4 Average availability and average restoration overbuild under DPP and SPP

Figure of Merit	DfnGwin	Germany50	Cost266	KL	Janos-US-CA	NSFnet
DPP - Conn. Availability	0.999983	0.999985	0.999790	0.999838	0.999723	0.999620
DPP - Restoration Overbuild	2.45	2.78	2.70	2.65	2.73	2.96
SPP - Conn. Availability	0.999932	0.999884	0.998272	0.999483	0.996525	0.998842
SPP - Restoration overbuild	1.69	1.85	2.02	2.24	2.06	2.30

Another interesting result concerns the *average minimum distance*, evaluated in hops between node pairs. For topologies with a small average minimum distance, a short total hop count can be expected in both DPP and SPP, yielding a suitable availability value. However, other factors such as link length distribution and sharing rules can modify the expected values. It has also been observed in SPP that the sharing group size can change the expected availability values if only topology features are considered. With respect to average node degree, it can be noted that under DPP it improves restoration overbuild because it helps in finding disjoint paths for backup.
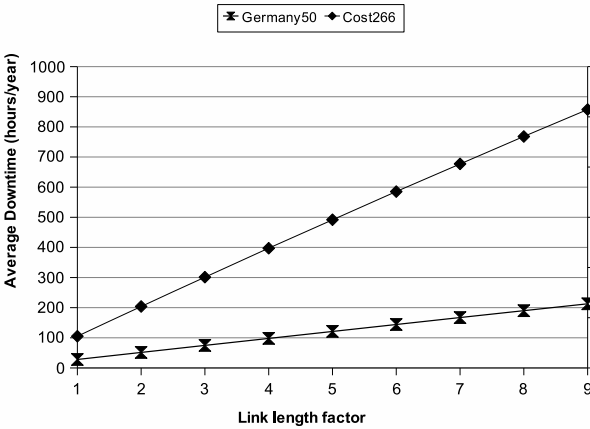
(a) Unprotected connections
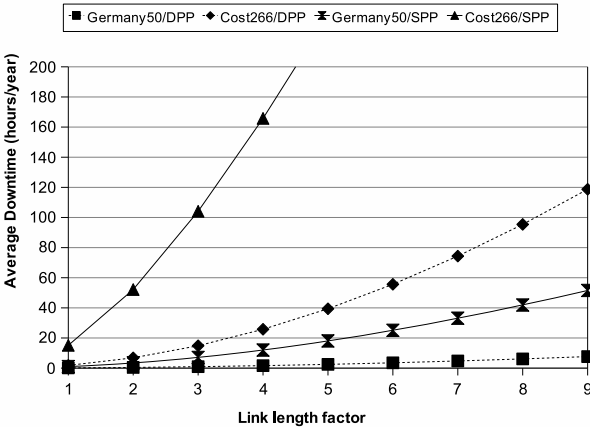
(b) Protected with DPP and SPP

Fig. 5.7 Average downtime as a function of scaled link length

References

1. Andersen, R., Chung, F., Sen, A., Xue, G.: On disjoint path pairs with wavelength continuity constraint in WDM networks. In: INFOCOM (2004)
2. Autenrieth, A., Kirstädter, A.: Engineering end-to-end ip resilience using resilience-differentiated QoS. IEEE Communications Magazine **40**(1), 50–57 (2002). DOI 10.1109/35.978049
3. Batchelor, P., Daino, B., Heinzmann, P., Hjelme, D., Inkret, R., Jager, H., Joindot, M., Kuchar, A., Coquil, E., Leuthold, P., Marchis, G., Matera, F., Mikac, B., Nolting, H. P., Spath, J., Tillerot, F., Caenegem, B., Wauters, N., Weinert, C.: Study on the implementation of optical transparent transport networks in the european environment-results of the research project cost 239. Photonic Network Communications **2**, 15–32(18) (March 2000)
4. Berger, L., Bryskin, I., Papadimitriou, D., Farrel, A.: GMPLS segment recovery. IETF RFC 4873 (2007)
5. Chołda, P.: The Reliability Analysis of Recovery Procedures in GMPLS-based Optical IP Networks. Ph.D. thesis, AGH University of Science and Technology, Kraków, Poland (2005). URL http://kt.agh.edu.pl/~cholda/Papers/PhD.pdf. Submitted
6. Cholda, P., Mykkeltveit, A., Helvik, B., Wittner, O., Jajszczyk, A.: A survey of resilience differentiation frameworks in communication networks. Communications Surveys and Tutorials, IEEE **9**(4), 32–55 (2007). DOI 10.1109/COMST.2007.4444749
7. Doucette, J., Grover, W. D.: Comparison of mesh protection and restoration schemes and the dependency on graph connectivity. In: Design of Reliable Communication Networks, pp. 121–128 (2001)
8. Doucette, J., He, D., Grover, W., Yang, O.: Algorithmic approaches for efficient enumeration of candidate p-cycles and capacitated p-cycle network design. Design of Reliable Communication Networks, 2003. (DRCN 2003). Proceedings. Fourth International Workshop on pp. 212–220 (2003). DOI 10.1109/DRCN.2003.1275359
9. Ellinas, G., Bouillet, E., Ramamurthy, R., Labourdette, J. F., Chaudhuri, S., Bala, K.: Routing and restoration architectures in mesh optical networks. Optical Networks Magazine pp. 91–106 (2003)
10. Feldman, J., Ruhl, M.: The directed steiner network problem is tractable for a constant number of terminals. In: Proc. 40th annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 299–308 (1999)
11. Grover, W., Doucette, J., Clouqueur, M., Leung, D., Stamatelakis, D.: New options and insights for survivable transport networks. IEEE Communications Magazine **40**(1), 34–41 (2002)
12. Gummadi, K. P., Pradeep, M. J., Murthy, C. S. R.: An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks. IEEE/ACM Transactions on Networking **11**(1), 81–94 (2003)
13. Haider, A., Harris, R.: Recovery techniques in next generation networks. Communications Surveys and Tutorials, IEEE **9**(3), 2–17 (2007). DOI 10.1109/COMST.2007.4317617
14. Ho, P. H., Tapolcai, J., Mouftah, H. T.: On optimal diverse routing for shared protection in mesh WDM networks. IEEE Transactions on Reliability **53**(6), 2216–225 (2004)
15. Hu, J. Q.: Diverse routing in optical mesh networks. IEEE Transactions on Communications **51**, 489–494 (2003)
16. Hülsermann, R., Jäger, M., Koster, A., Orlowski, S., Wessäly, R., Zymolka, A.: Availability and cost based evaluation of demand-wise shared protection. In: 7th ITG-Workshop on Photonic Networks, Leipzig, Germany, pp. 161–168. VDE Verlag GmbH (2006)
17. Iraschko, R. R., Grover, W. D.: A highly efficient path-restoration protocol for management of optical network transport integrity. IEEE Journal on Selected Areas in Communications (JSAC) **18**(5), 779–794 (2000)
18. Jäger, M., R. Hüelsermann, D. A. S., Sedlak, R.: Evaluation of novel resilience schemes in dynamic optical transport networks. In: Proc. APOC (2002)

19. Laborczi, P., Tapolcai, J., Ho, P. H., Cinkler, T., Recski, A., Mouftah, H. T.: Algorithms for asymmetrically weighted pair of disjoint paths in survivable networks. In: Design of Reliable Communication Networks, pp. 220–227 (2001)
20. Lai, W., McDysan, D.: Network hierarchy and multilayer survivability. IETF RFC 3386 (2002)
21. Liu, Y., Tipper, D.: Spare capacity allocation for non-linear link cost and failure-dependent path restoration. In: Design of Reliable Communication Networks, pp. – (2001)
22. Liu, Y., Tipper, D., Siripongwutikorn, P.: Approximating optimal spare capacity allocation by successive survivable routing. In: INFOCOM, pp. 699–708. Anchorage, Alaska (2001)
23. Mohan, G., Murthy, C. S. R.: Lightpath restoration in WDM optical networks. IEEE Network **14**(6), 24–32 (2000)
24. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0–Survivable Network Design Library. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007)
25. Ou, C. S., Zang, H., Mukherjee, B.: Sub-Path Protection for Scalability and Fast Recovery in WDM Mesh Networks. In: Proc. Optical Fiber Communications Conference (OFC), p. ThO6 (2002)
26. Pan, P., Swallow, G., Atlas, A.: Fast reroute extensions to RSVP-TE for LSP tunnels. IETF RFC 4090 (2001)
27. Ramamurthyt, S., Mukherjee, B.: Survivable WDM mesh networks, part II – restoration. In: International Conference on Communications, pp. 2023–2030 (1999)
28. Ramasubramanian, S.: On failure dependent protection in optical grooming networks. In: International Conference on Dependable Systems and Networks, pp. 440–449 (2004)
29. Segovia, J., Calle, E., Vila, P., Marzo, J., Tapolcai, J.: Topology-focused availability analysis of basic protection schemes in optical transport networks. J. Opt. Netw. **7**(4), 351–364 (2008)
30. Optical fiber supply meets expanding application. SK Opto-Electronics Newsletter (2003)
31. Stidsen, T., Kjaerulff, P.: Complete rerouting protection. Journal of Optical Networking **5**(6), 481–492 (2006)
32. Tapolcai, J., Cinkler, T.: On-line routing algorithms with shared protection in WDM networks. In: Optical Network Design and Modelling, pp. 351–364 (2003)
33. Tapolcai, J., Ho, P. H., Cinkler, T.: A compact mathematical formulation for shared path protection with general shared risk groups. In: Conference on Wireless and Optical Communications Networks (WOCN), pp. 270–274. Dubai, UAE (2005)
34. Tapolcai, J., Ho, P. H., Verchere, D., Cinkler, T.: A novel shared segment protection method for guaranteed recovery time. In: Broadband Optical Networking Symposium (BroadNets), pp. 127–136 (2005)
35. Verbrugge, S., Colle, D., Demeester, P., Hülsermann, R., Jäger, M.: General availability model for multilayer transport networks. In: Proc. DRCN 2005. Lacco Ameno, Italy (2005)
36. Wang, D., Li, G.: Efficient distributed solution for MPLS Fast Reroute. In: 4th International IFIP-TC6 Networking Conference (NETWORKING), p. 804 (2005)
37. Wang, H., Modiano, E., Médard, M.: Partial path protection for wdm networks: End-to-end recovery using local failure information. In: Seventh International Symposium on Computers and Communications (ISCC), pp. 719–725 (2002)
38. Xiong, Y., Mason, L.: Restoration strategies and spare capacity requirements in self-healing ATM networks. IEEE/ACM Transactions on Networking **7**(1), 98–110 (1999)
39. Xu, D., Xiong, Y., Qiao, C., Li, G.: Trap avoidance and protection schemes in networks with shared risk link groups. In: IEEE Journal of Lightwave Technology (2003)

Chapter 6
Routing Optimization in Optical Burst Switching Networks: a Multi-path Routing Approach

Mirosław Klinkowski, Marian Marciniak, and Michał Pióro

Abstract This chapter concerns routing optimization in optical burst switching (OBS) networks. OBS is a photonic network technology aiming at efficient transport of IP traffic. OBS architectures are in general bufferless and therefore sensitive to burst congestion. An overall burst loss probability (BLP) which adequately represents the congestion state of the entire network is the primary metric of interest in an OBS network. The network congestion can be reduced by using proper routing. We consider multi-path source routing and aim at optimal distribution of traffic over the network. In this context, we study three network loss models, a well-known loss model of an OBS network and two original approximate models. Since the objective function of each model is nonlinear, either linear programming formulations with piecewise linear approximations of this function or nonlinear optimization gradient methods can be used. The presented solution is based on nonlinear optimization; for this purpose we provide the formulas for calculation of partial derivatives. The main goal of this chapter is to show that the use of approximate models allows us to speed up significantly the optimization procedure without losing much accuracy. Moreover we show that our method effectively distributes the traffic over the network, and the overall BLP can be reduced compared with both shortest path routing and alternative routing.

Mirosław Klinkowski

Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya, Barcelona, Spain, and

Department of Transmission and Optical Technology, National Institute of Telecommunications, Warsaw, Poland, e-mail: `mklinkow@ac.upc.edu`

Marian Marciniak

Department of Transmission and Optical Technology, National Institute of Telecommunications, Warsaw, Poland, e-mail: `m.marciniak@itl.waw.pl`

Michał Pióro

Institute of Telecommunications, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland, and

Department of Electrical and Information Technology, Lund University, Box 118, SE-221 00 Lund, Sweden, e-mail: `mpp@tele.pw.edu.pl`

Key words: multi-path routing, network optimization, nonlinear optimization, optical burst switching

6.1 Introduction

Optical Burst Switching (OBS) is a photonic network technology aiming at efficient transport of IP traffic [20]. OBS architectures are in general bufferless and as such are sensitive to burst congestion. An overall burst loss probability (BLP) which adequately represents the congestion state of the entire network is the primary metric of interest in an OBS network. The *network congestion* can be reduced by using proper routing; in this context alternative (or deflection) routing (e.g., [1]), a common routing strategy in OBS, has been considered. Although deflection routing improves network performance under low traffic loads, it may still increase burst losses under moderate and high loads.

In this chapter we consider another approach – *multi-path source routing* – and use network optimization theory to distribute the traffic in an optimal way. This work completes and extends our previous works [11] and [12]. We investigate three different *network loss models*, a well-known loss model of an OBS network [21] and two approximate models developed by us. As the cost function, which represents the overall burst loss probability, is nonlinear, either linear programming formulations with piecewise linear approximations of this function [22] or *nonlinear optimization gradient methods* [7] can be used. We make use of the latter approach.

In our nonlinear optimization problem we assume that there is a preestablished virtual path topology consisting of a limited number of paths between each pair of source-destination nodes. Using a gradient optimization method we calculate a traffic splitting vector that determines the distribution of traffic over these paths. In order to support the gradient method we provide straightforward formulas for calculation of partial derivatives. The main goal of this chapter is to show that the use of approximate models allows us to speed up significantly the optimization procedure without losing much accuracy. Moreover, we show that our method effectively distributes the traffic over the network, and the overall BLP can be reduced compared with both shortest path routing and alternative routing. The proposed solution can be used, in particular, for static (preplanned) multi-path source routing, where the traffic distribution is calculated based on a given (long-term) traffic demand matrix. Then, either a periodic or a threshold-triggered update of the splitting vector can be performed if the traffic demand matrix is subject to a change.

The chapter is structured as follows. In Section 6.2 we provide a description of OBS technology and briefly review routing methods considered for OBS networks. In Section 6.3 we discuss OBS network loss models and introduce a multi-path source routing model. In Section 6.4, for each of the introduced network loss models, we formulate the objective function, calculate the partial derivatives, and present some numerical results. In Section 6.5 we investigate the accuracy of network loss models and the characteristics of the objective function, and discuss the

computational effort of the optimization method. Finally, Section 6.6 contains the conclusions.

6.2 OBS Technology

OBS technology is a promising solution for reducing the gap between transmission and switching speed in future networks. The principal design objective for an OBS network is that the aggregated user data, called the burst, be carried transparently through the network as an optical signal, i.e., without any optical-to-electrical conversion. This optical signal passes through the switches that have either none or very limited buffering capabilities. The control information is carried on a dedicated wavelength and separately from the user data. This information is delivered to switching nodes with some offset time, prior to the data burst, so that the node can process it and set up the switching matrix in advance. In such a network the wavelength resources are allocated temporarily and shared between different connections. Such an operation increases network flexibility and adaptability to the bursty characteristics of IP traffic. Moreover, the aggregation of user data helps to reduce the scale of control information processed in the network and it relaxes the switching requirements. Since the control information and the user data are separated, they can be encoded with different modulation formats and transmitted at different rates. Such division improves network management and provides additional flexibility.

A conventional OBS network operates with a one-way signalling mode and it allocates transmission resources on the fly, a while before the burst arrives to the node. Since there is no acknowledgement about the availability of network resources, it may happen that two bursts want to access the same wavelength resources at the same time. The problem of such a burst contention is crucial in OBS networks. The conversion of wavelength is a natural mechanism used to solve this problem [4]. In this mechanism, the carrier frequency of a contending optical signal is converted to another available one. Deflection (or alternative) routing is another contention resolution mechanism considered for OBS networks. In this case, a contending burst is forwarded spatially, in the switching matrix, to another output port (fiber).

6.2.1 Routing Methods

Static shortest path routing based on Dijkstra's algorithm is the primary routing method frequently explored in OBS networks (e.g., [24]). Such routing reduces overall network utilization when calculated with respect to the number of hops. On the other hand, some links may be overloaded, while others may be spare, leading to excessive burst losses. Therefore, several reactive and proactive routing strategies, based on alternative, multi-path, and single-path routing, have been proposed with the objective of the reduction of burst congestion.

Although alternative routing can improve the network performance under low traffic load conditions, it may still intensify the burst losses under moderate and high loads [25]. Indeed the general problem of alternative routing in bufferless OBS networks is the overutilization of link resources, which happens if an alternative path has more hops than a primary path. Hence, whereas early proposals were based on static route calculation and selection (e.g., [8]), as a next step some authors proposed an optimized calculation of the set of alternative routes [15] as well as an adaptive selection of paths [2]. The assignment of lower priorities to deflected bursts is another important technique that protects against excessive burst losses on primary paths [1].

Multi-path routing represents another group of routing strategies which aim at traffic load balancing in OBS networks. Most of the proposals are based on a static calculation of the set of equally important routes, usually with the Dijkstra algorithm. Then the path selection is performed adaptively and according to some heuristic [18] or optimized cost function [22] [16]. Both traffic splitting [14] and path ranking [23] techniques are used in the path selection process.

The network congestion in single-path routing can be avoided thanks to a proactive route calculation. Although most of the strategies proposed for OBS networks consider centralized calculation of single routes [22], some authors still focus on distributed routing algorithms [5]. Both optimization [26] and heuristic [3] methods are used.

6.3 Network Modeling

We use $G = (V, E)$ to denote the graph of an OBS network; the set of nodes is denoted as V, and the set of links is denoted as E. Link $e \in E$ comprises C_e wavelengths. $\mathscr{P}$ denotes the set of paths predefined between source s and destination t nodes, $s, t \in V$, and $s \neq t$. Each individual path $p \in \mathscr{P}$ is identified with a subset $p \subseteq E$. Subset $\mathscr{P}_{st} \subseteq \mathscr{P}$ identifies all paths from source s to destination t; the sets $\mathscr{P}_{st}$ are disjoint in our model. Subset $\mathscr{P}_e \subseteq \mathscr{P}$ identifies all paths that go through link e.

The reservation (holding) times on each link are independent and identically distributed random variables with the mean equal to the mean burst duration h; for simplicity we assume $h = 1$. We assume that the network is capable of full wavelength conversion, i.e., a burst can be transmitted on any available wavelength in each link. The demand traffic pattern is described by matrix $[\gamma_{st}]_{s,t \in V}$ and bursts destined to given node t arrive at node s according to a Poisson process of (long-term) rate $\gamma_{st}/h = \gamma_{st}$.

Later we use ρ_p and ρ_e to denote the traffic offered to path $p \in \mathscr{P}$ and the traffic offered to link $e \in E$, respectively.

In the following two subsections we deal with the modeling of the volume of burst traffic lost in the OBS network. The procedure consists, in the first step, of the calculation of burst loss probabilities E_e on individual links, and in the second step,

of the calculation of BLP in the entire network. Finally, we introduce a multi-path source routing model.

6.3.1 Link Loss Calculation

By assuming the network has full wavelength conversion capability, i.e., each wavelength can be selected whenever it is available, the blocking probability E_e on each link is given by the following Erlang loss formula (see [21]):

$$E_e = E(\rho_e, C_e) = \frac{\rho_e^{C_e}}{C_e!} \left[\sum_{i=0}^{C_e} \frac{\rho_e^i}{i!} \right]^{-1}, \quad e \in E. \tag{6.1}$$

In order to determine E_e, $e \in E$, we have to calculate the traffic load ρ_e offered to individual links; recall that C_e, $e \in E$ is given. Below, we provide two models of such a calculation.

Reduced load (RL). A common loss model of an OBS network was proposed by Rosberg et al. [21] and it makes use of a *reduced load* calculation. This model is an extension of the model proposed by Kelly [9] for circuit-switching (CS) networks. In the OBS network, it is assumed that the traffic offered to link e is obtained as a sum of the traffic offered to all the paths that cross this link reduced by the traffic lost in the preceding links along these paths.

This relation can be expressed as

$$\rho_e = \sum_{p \in \mathscr{P}_e} \rho_p \Lambda_{pe}, \quad e \in E, \tag{6.2}$$

where

$$\Lambda_{pe} = \prod_{f \in r_{pe}} (1 - E_f), \quad p \in \mathscr{P}, e \in E, \tag{6.3}$$

and subset $r_{pe} \subset p$ identifies all links that precede link e along path p.

The difference between this model and the CS network model is that in the latter the subset r_{pe} contains all the links that succeed link e along path p, on top of all preceding links. This difference reflects the fact that a burst offered to path p in OBS uses a single wavelength from each link along the path until the first link where it is being blocked or until it exists in the network. In contrast, a connection in CS either occupies a channel in all the links along the path or is blocked.

The calculation of link loss probabilities E_e, $e \in E$, together with the calculation of offered burst traffic ρ_e, given by the reduced load model (6.2), leads to a fixed-point equation with a solution known as the *Erlang fixed point*. The fixed point cannot be solved in a closed form but its approximation can be found through repeated substitution of (6.1) in (6.2). It is known that the fixed point exists in both
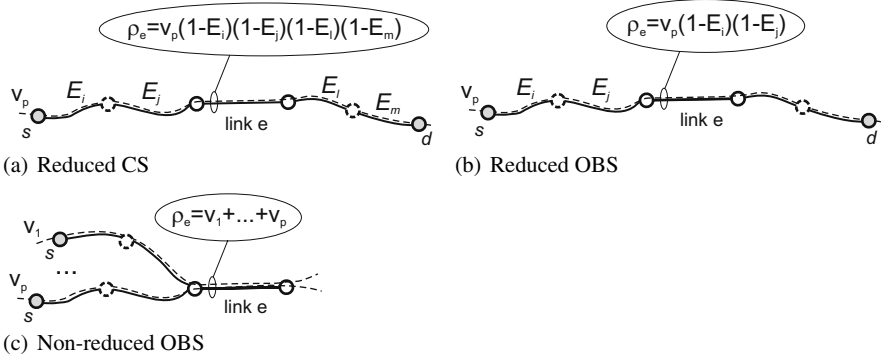
(a) Reduced CS

(b) Reduced OBS

(c) Non-reduced OBS

Fig. 6.1 Link load models

CS and OBS networks (see [9] and [21], respectively). Although the fixed point is unique in CS networks, its uniqueness has not been proved in OBS networks.

Although the traffic offered to a route is Poisson, it may still be thinned by blocking at consecutive links and thus no longer remain Poisson. Since there is no straightforward solution to this problem, we make a simplification that the burst arrival process to each link is Poisson.

Non-reduced load (NRL). Formulation (6.2) may bring some computational difficulty, especially with regard to the calculation of partial derivatives for optimization purposes. Therefore, we also consider a simplified *non-reduced load* model, where the traffic offered to link e is calculated as a sum of the traffic offered to all paths that cross this link:

$$\rho_e = \sum_{p \in \mathscr{P}_e} \rho_p, \quad e \in E. \tag{6.4}$$

The rationale behind this assumption is that under low link losses E_f, $f \in E$, observed in a properly dimensioned network, model (6.2) can be approximated by (6.4).

Figure 6.1 presents illustrative examples of the reduced load calculation for both CS and OBS networks, as well as of the non-reduced load calculation.

6.3.2 Network Loss Calculation

Overall network loss (NL). The calculation of overall burst loss or blocking probability in an OBS network is presented in [21], and it uses the same formulation as was proposed for CS networks [9]. In further discussion we name this model an overall network loss (NL) model.

The main modeling steps include the calculation of

1. burst loss probabilities E_e on links, given by (6.1),
2. loss probabilities L_p of bursts offered to paths

$$L_p = 1 - \prod_{e \in p} (1 - E_e), \quad p \in \mathcal{P}, \text{and} \tag{6.5}$$

3. the overall burst loss probability B_{NL},

$$B_{NL} = \sum_{p \in \mathcal{P}} \rho_p L_p \left[\sum_{p \in \mathcal{P}} \rho_p \right]^{-1}. \tag{6.6}$$

In order to calculate the path loss probability L_p, $p \in \mathcal{P}$, we make an assumption that burst blocking events occur independently at the network links. Then formula (6.5) accounts for blocking probabilities in all links e that belong to path p.

The overall burst loss probability B_{NL} is calculated simply as the volume of burst traffic lost in the network normalized to the volume of burst traffic offered to the network.

Overall link loss (LL). Another method for calculation of burst losses in the entire network is based on an overall link loss (LL) model [6]. In this method we sum up the volumes of traffic lost on individual network links.

The main modeling steps include the calculation of

1. burst loss probabilities E_e on links, given by (6.1), and
2. B_{LL}, a sum of the burst traffic lost on individual links relative to the overall traffic offered to the network

$$L_{LL} = \sum_{e \in E} \rho_e E_e \left[\sum_{p \in \mathcal{P}} \rho_p \right]^{-1}. \tag{6.7}$$

LL overestimates actual burst losses given by (6.6) in NL because it counts twice the intersection of blocking events that occur on distinct links. In fact, B_{LL} may be higher than 1, and thus it cannot be considered as the probability metric. Nevertheless, for $E_e \to 0$, $e \in E$, the blocking events that occur simultaneously vanish rapidly, and model (6.7) converges to model (6.6).

6.3.3 Multi-path Source Routing

We assume that the network applies source-based routing, so that the source node determines the path of a burst that enters the network (see Figure 6.2). Moreover, the network uses multi-path routing where each subset $\mathcal{P}_{st}$ comprises a (small) number of paths, and a burst can follow one of them. We assume that the selection of a route from set $\mathcal{P}_{st}$ is random for each burst and is performed according to a given traffic splitting factor x_p, such that
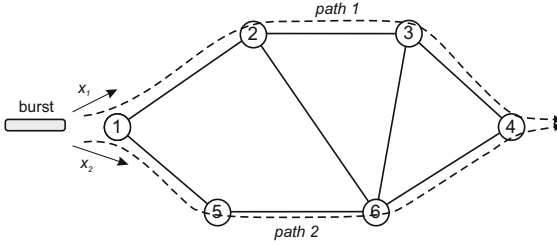
Fig. 6.2 An example of OBS network with source-based routing; x_1 and x_2 are the traffic splitting factors and $x_1 + x_2 = 1$

$$0 \leq x_p \leq 1 \qquad\qquad p \in \mathscr{P}, \qquad\qquad (6.8\text{a})$$

$$\sum_{p \in \mathscr{P}_{st}} x_p = 1 \qquad\qquad s,t \in V, s \neq t. \qquad\qquad (6.8\text{b})$$

Thus traffic ρ_p offered to path $p \in \mathscr{P}_{st}$ can be calculated as

$$\rho_p = x_p \tau_p, \qquad\qquad (6.9)$$

where $\tau_p = \gamma_{st}$ is the total traffic offered between s and t.

Here vector $x = (x_1, \ldots, x_{|\mathscr{P}|})$ determines the distribution of traffic over the network; this vector should be optimized to reduce congestion and to improve overall performance.

6.4 Resolution Methods and Numerical Examples

6.4.1 Formulation of the Optimization Problem

Taking into account different methods of the link load and the network loss calculation presented in Section 6.3, several network loss models with corresponding objective functions can be defined.

1. **NL-RL**. The link load is calculated according to the RL model given by (6.2), and the network loss is calculated according to the NL model given by (6.6), with the objective function given by

$$B_{NL-RL}(x) = \sum_{p \in \mathscr{P}} x_p \tau_p L_p. \qquad\qquad (6.10)$$

2. **NL-NRL**. The link load is calculated according to the NRL model given by (6.4), and the network loss is calculated according to the NL model given by (6.6), with the objective function given by

$$B_{NL-NRL}(x) = \sum_{p \in \mathscr{P}} x_p \tau_p L_p. \tag{6.11}$$

3. **LL-NRL**. The link load is calculated according to the NRL model given by (6.4), and the network loss is calculated according to the LL model given by (6.7), with the objective function given by

$$B_{LL-NRL}(x) = \sum_{e \in E} \rho_e E_e = \sum_{e \in E} E_e \left(\sum_{p \in \mathscr{P}_e} x_p \tau_p \right). \tag{6.12}$$

The last possible combination of the link load and the network loss calculation is LL-RL. Because such a model does not bring much gain with respect to the NL-RL model, as it does not avoid the complexity of fixed-point calculation, we do not study it.

In each case the normalization factor $\left[\sum_{p \in \mathscr{P}} \rho_p \right]^{-1}$ has been omitted because we assume it to be a constant value.

The optimization problem is the same for each method, and is formulated as follows:

$$\min_x B(x) \tag{6.13}$$

subject to the multi-path routing constraints given by (6.8a) and (6.8b).

Since in each case $B(x)$ is a nonlinear function of vector x, the optimization problem is nonlinear. Taking into account the form of both constraints (6.2) and (6.4), a particularly convenient optimization method is the Frank-Wolfe reduced gradient method (algorithm 5.10 in [19]); this algorithm was used for a similar problem in circuit-switched (CS) networks [7].

6.4.2 Calculation of Partial Derivatives

In general, gradient methods are iterative methods used in the optimization of convex functions. Gradient methods need to employ the calculation of partial derivatives of the cost function to find the direction for its improvement. Below, we provide adequate formulas for the partial derivatives for each of the models.

NL-RL model. The partial derivative of B_{NL-RL} with respect to $x_q, q \in \mathscr{P}$, can be derived directly by a standard method involving the solution of a system of linear equations. It follows from (6.2) and (6.1) that

$$\frac{\partial \rho_e(x)}{\partial x_q} = \alpha_{qe} \tau_q \Lambda_{qe} + \sum_{p \in \mathscr{P}_e} x_p \tau_p \Lambda_{pe} \sum_{f \in r_{pe}} (1 - E_f)^{-1} \frac{\partial E_f(x)}{\partial x_q}, \quad e \in E, q \in \mathscr{P}, \tag{6.14}$$

where $\alpha_{qe} = 1$ if $e \in q$, and $\alpha_{qe} = 0$ otherwise, and

$$\frac{\partial E_e(x)}{\partial x_q} = E_e \left(E_e + \frac{C_e - \rho_e}{\rho_e} \right) \frac{\partial \rho_e(x)}{\partial x_q}, \quad e \in E, q \in \mathscr{P}. \tag{6.15}$$

In order to solve the system of equations (6.14)–(6.15), a fixed-point calculation procedure, i.e., repeated substitution of (6.14) in (6.15), has to be applied.

From (6.5) we have

$$\frac{\partial L_p(x)}{\partial x_q} = (1 - L_p) \sum_{e \in p} (1 - E_e)^{-1} \frac{\partial E_e(x)}{\partial x_q}, \quad p, q \in \mathscr{P}, \tag{6.16}$$

and finally from (6.10),

$$\frac{\partial}{\partial x_q} BNL - RL(x) = \tau_q L_q + \sum_{p \in \mathscr{P}} x_p \tau_p \frac{\partial L_p(x)}{\partial x_q}, \quad q \in \mathscr{P}. \tag{6.17}$$

The calculation of partial derivatives (6.14)–(6.17) in NL-NRL model is extremely time consuming since it involves an iterative fixed-point approximation procedure.

NL-NRL model. The partial derivative of B_{NL-NRL} with respect to x_q, $q \in \mathscr{P}$, could be derived directly from formulas (6.1) and (6.4)–(6.6) by a standard method involving resolution of a system of linear equations, similarly to (6.14)–(6.17). Although there is no need for a fixed-point calculation in NL-NRL model, still such a computation would be time consuming.

Therefore, we propose instead a straightforward exact calculation based on the approach for CS networks by Kelly [10]; a detailed derivation of formulas is presented in [11]. In particular, for each path $q \in \mathscr{P}$ we have

$$\frac{\partial}{\partial x_q} B_{NL-NRL}(x) = \tau_q \left[L_q + \sum_{e \in q} c_e \right], \tag{6.18}$$

where c_e is calculated for each link $e \in E$ as

$$c_e = \eta_e \sum_{p \in \mathscr{P}_e} \rho_p (1 - L_p), \tag{6.19}$$

and

$$\eta_e = E(\rho_e, C_e - 1) - E(\rho_e, C_e), \quad e \in E. \tag{6.20}$$

Due to assumption (6.4) we have managed to simplify the model (6.2) and make the calculation of partial derivatives defined by (6.18) and (6.19) straightforward, not involving any iterations. Indeed, once $|E|$ of unknowns (c_e) are pre-calculated they can be used in (6.18) to obtain the partial derivatives. Calculating the gradient in this method, therefore, is not longer an issue.
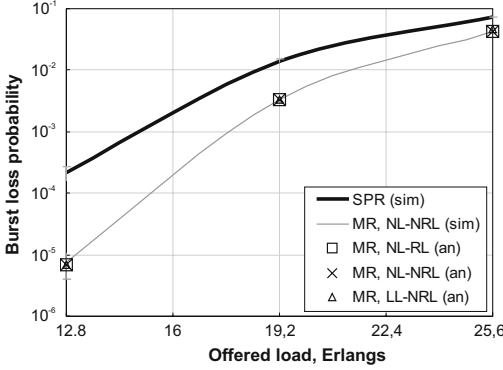
Fig. 6.3 Validation of optimization models

LL-NRL model. The partial derivative of B_{LL-NRL} with respect to x_q, $q \in \mathscr{P}$, can be derived directly from formulas (6.1), (6.4), and (6.12),

$$\frac{\partial}{\partial x_q} B_{LL-NRL}(x) = \tau_q \left[\sum_{e \in q} (E_e + \eta_e \rho_e (1 - E_e)) \right], \tag{6.21}$$

where η_e is given by equation (6.20).

6.4.3 Numerical Results

We evaluated the performance of our multi-path source routing scheme in an event-driven simulator. In order to find a splitting vector x specifying near-optimal routing we used a solver `fmincon` for constrained nonlinear multivariable functions available in the MATLAB optimization toolbox. Then we applied this vector in the simulator.

The evaluation was performed for NSFNET, an American backbone network topology of 15 nodes and 23 links [17]; each link had $C = 32$ wavelengths and the transmission bitrate in each wavelength channel was 10 Gbit/s. Besides the results of optimized multi-path routing (MR) we provide, for comparison, the results of two other routing strategies: simple shortest path routing (SPR) and pure alternative routing (AR). We considered two shortest paths per source-destination pair of nodes in MR; they were not necessarily disjoint. In SPR only one path was available, while in the case of AR we considered two different scenarios: with two and six paths available. Uniform traffic matrix and exponential burst inter-arrivals and durations were considered. All the simulation results had 99% level of confidence.

In Figure 6.3 we show the overall burst loss probability results of the MR strategy, which was optimized with the assistance of NL-NRL, NL-RL, and LL-NRL models,
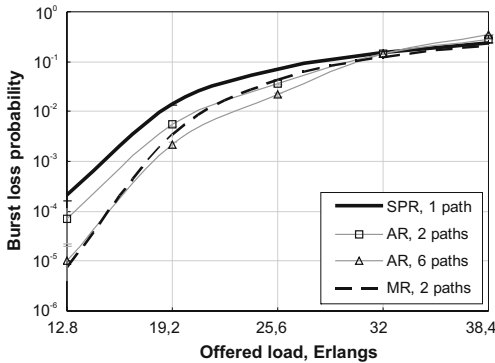
Fig. 6.4 Performance comparison of routing strategies (simulation results)

successively. The characteristics are obtained in the function of offered traffic load, which is normalized to the wavelength bitrate and expressed in *Erlangs* (e.g., 12.8 Erlangs means that each node generates 128 Gbit/s). As a reference, we provide the results of SPR.

In the studied scenario, we can see that the burst loss probability results of optimized MR evaluated in the MATLAB environment are (almost) the same regardless of the network loss model used. Moreover, the analytical results obtained for NL-NRL model agree very well with simulation results ('(sim)' in Figure 6.3).

In Figure 6.4 we compare simulation results obtained for different routing scenarios. We see that the optimized multi-path routing outperforms the shortest path routing in the whole range of traffic loads. Also, it offers at least as good results as the alternative routing if the same number of routing paths is available.

6.5 Discussion

In this section we investigate the accuracy of network loss models and the characteristics of the objective function. We also discuss the computational effort of the optimization procedure.

6.5.1 Accuracy of Loss Models

We study the accurary of both NR-NRL and LL-NRL network loss approximations relative to the NL-RL network loss model. To do that we define the approximation error as
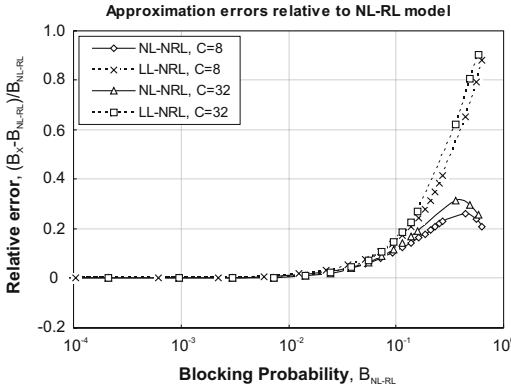
Fig. 6.5 Approximation errors relative to NL-RL model vs. blocking probability; NSFNET, shortest path routing, eight and 32 wavelengths

$$Er_X \triangleq \frac{B_X - B_{NL-RL}}{B_{NL-RL}}, \tag{6.22}$$

where X refers to either $NL - NRL$ or $LL - NRL$; so B_X means the result of the objective function for model X.

In Figure 6.5 we present the results of Er_X obtained in NSFNET network, with a different number of wavelengths per link considered and the shortest path routing used. We can see that the accuracy of both network loss approximate models is very strict for the blocking probability in the network B_{NL-RL} below 10^{-2}.

6.5.2 Properties of the Objective Function

NL-RL model. In [10], Kelly demonstrated that the reduced load loss model of a CS network is in general not convex. Taking into account an analogy of the reduced-load calculation in both CS and OBS networks, we can expect that function (6.10) is not convex as well. Therefore, a solution of optimization problem (6.13) may not be unique.

NL-NRL model. As in the case of the RL-NL model, it can be shown numerically that the objective function (6.11) is not necessarily convex; in particular, under high traffic load conditions, two feasible vectors $\mathbf{x}_1$, $\mathbf{x}_2$ can be found such that

$$B_{NL-NRL}(\lambda \mathbf{x}_2 + (1-\lambda)\mathbf{x}_2) > \lambda B_{NL-NRL}(\mathbf{x}_2) + (1-\lambda)B_{NL-NRL}(\mathbf{x}_1), \tag{6.23}$$

where $0 \leq \lambda \leq 1$.

Table 6.1 Comparison of computation times

				NL-RL		NL-NRL		LL-NRL	
Network	Paths	Tol.	BLP	OF	SOLV	OF	SOLV	OF	SOLV
SIMPLE	2	10^{-6}	$2.4 \cdot 10^{-3}$	64 sec	1.5 sec	0.1 sec	1.4 sec	0.1 sec	1.5 sec
SIMPLE	4	10^{-6}	$2.4 \cdot 10^{-3}$	243 sec	3 sec	0.1 sec	3.4 sec	0.1 sec	3.1 sec
NSFNET	2	10^{-6}	$4.6 \cdot 10^{-2}$	$> 5h$		0.38 sec	22.3 sec	0.37 sec	24.3 sec
NSFNET	4	10^{-6}	$3.1 \cdot 10^{-2}$	$> 5h$		1.6 sec	937 sec	1.5 sec	952 sec
EON	2	10^{-6}	$1.76 \cdot 10^{-2}$	$> 5h$		5.5 sec	803 sec	5.3 sec	837 sec
EON	2	10^{-3}	$1.77 \cdot 10^{-2}$	$> 5h$		1.1 sec	260 sec	1.0 sec	263 sec

LL-NRL model. An advantageous property of the LL-NRL model is the convexity of its objective function (6.12); a detailed proof can be found in [13]. For this reason, a corresponding optimization problem has a unique solution.

6.5.3 Computational Effort

In Table 6.1 we compare the computation times of both the objective function (with the partial derivatives calculation included) and the fmincon solver function of the MATLAB environment; in the table they are denoted as OF and SOLV, respectively. The evaluation is performed on a Pentium D, 3 GHz computer. The results are obtained for SIMPLE (six nodes, eight links, and 60 paths), NSFNET (15 nodes, 23 links, and 420 paths), and EON (28 nodes, 39 links, and 1,512 paths) network topologies; the number of wavelengths per link is 32, each source-destination pair of nodes has two or four shortest paths available, the traffic load is equal to 25.6 Erlangs and 19.2 Erlangs, respectively, for SIMPLE/NSFNET and EON scenarios. In case the iterative procedure of the Erlang fixed-point approximation is used, it ends if the maximal discrepancy between two consecutive link loss calculations is smaller then 10^{-6}. The starting traffic splitting vector is $\mathbf{x} = 0.5 \cdot (1, ..., 1)$, meaning that the traffic is equally distributed on the paths for each demand.

We can see that the calculation of the objective function (and of partial derivatives) is highly time consuming in the NL-RL model even in a small network scenario. In contrast, such a calculation is not an issue if either the NL-NRL or the LL-NRL model is used. It is worth noting that by decreasing the value of a termination tolerance parameter ('Tol.' in the table), which decides on the termination of the solver function, we significantly accelerate the optimization procedure (more than three times) without substantial decrease of routing performance (compare BLP value in both EON scenarios). Moreover, we can see that by increasing the number of paths the computation time of the solver function increases considerably in a larger (NSFNET) network scenario.

6.6 Conclusions

In this chapter we have studied a nonlinear optimization method for the multi-path source routing problem in OBS networks. In this method we calculate a traffic splitting vector that determines a near-optimal distribution of traffic over routing paths. Since a conventional network loss model of an OBS network is complex, we have introduced some simplifications. The proposed models are computationally effective and are still highly accurate compared to the basic model. The obtained formulas for partial derivatives are straightforward and very fast to compute. It makes the proposed nonlinear optimization method a viable alternative to linear programming formulations based on piecewise linear approximations of the cost function.

The simulation results demonstrate that our method effectively distributes the traffic over the network and the overall burst loss probability can be significantly reduced compared with the shortest path routing.

References

1. Cameron, C., Zalesky, A., Zukerman, M.: Prioritized deflection routing in optical burst switching networks. IEICE Trans. on Comm. **E88-B**(5), 1861–1867 (2005)
2. Coutelen, T., Elbiaze, H., Jaumard, B.: An efficient adaptive offset mechanism to reduce burst losses in OBS networks. In: Proceedings of Proceedings of IEEE Global Communications Conference (GLOBECOM 2005). St. Louis, MO (USA) (2005)
3. Du, Y., Pu, T., Zhang, H., Quo, Y.: Adaptive load balancing routing algorithm for optical burst-switching networks. In: Proceedings of Optical Fiber Communication Conference (OFC 2006). Anaheim, CL (USA) (2006)
4. Eramo, V., Listanti, M., Pacifici, P.: A comparison study on the number of wavelength converters needed in synchronous and asynchronous all-optical switching architectures. Journal of Lightwave Technology **21**(2), 340–355 (2003)
5. Gao, D., Zhang, H.: Information sharing based optimal routing for optical burst switching (OBS) network. In: Proceedings of Optical Fiber Communication Conference (OFC 2006). Anaheim, CL (USA) (2006)
6. Girard, A., Sanso, B.: Multicommodity flow models failure propagation, and reliable loss network design. IEEE/ACM Trans. on Net. **6**(1), 82–93 (1998)
7. Harris, R.: The modified reduced gradient method for optimally dimensioning telephone networks. Australian Telecom. Research **10**(1), 30–35 (1976)
8. Hsu, C., Liu, T., Huang, N.: Performance analysis of deflection routing in optical burst-switched networks. In: Proceedings of the 21st Joint Conference of IEEE Computer and Communications Societies (INFOCOM 2002). New York, NY (USA) (2002)
9. Kelly, F. P.: Blocking probabilities in large circuit-switched networks. Advanced Applied Probability **18**, 473–505 (1986)
10. Kelly, F. P.: Routing in circuit-switched networks: Optimization, shadow prices and decentralization. Advanced Applied Probability **20**, 112–144 (1988)

11. Klinkowski, M., Pioro, M., Careglio, D., Marciniak, M., Sole-Pareta, J.: Non-linear optimization for multi-path source routing in OBS networks. IEEE Communications Letters **11**(12) (2007)

12. Klinkowski, M., Pioro, M., Careglio, D., Marciniak, M., Sole-Pareta, J.: Routing optimization in optical burst switching networks. In: Proceedings of the 11th Conference on Optical Network Design and Modelling (ONDM 2007). Athens, Greece (2007)

13. Krishnan, K. R.: The convexity of loss rate in an Erlang loss system and sojourn in an Erlang delay system with respect to arrival and service rates. IEEE Trans. on Comm. **38**(9), 1314–1316 (1990)

14. Li, J., Mohan, G., Chua, K. C.: Dynamic load balancing in IP-over-WDM optical burst switching networks. Computer Networks **47**(3), 393–408 (2005)

15. Long, K., Yang, X., Huang, S., Chen, Q., Wang, R.: Adaptive parameter deflection routing to resolve contentions in OBS networks. In: Proceedings of the 5th International Conference on Networking (Networking 2006). Coimbra, Portugal (2006)

16. Lu, J., Liu, Y., Gurusamy, M., Chua, K.: Gradient projection based multi-path traffic routing in optical burst switching networks. In: Proceedings of IEEE High Performance Switching and Routing workshop (HPSR 2006). Poznan, Poland (2006)

17. NSFNET-the National Science Foundation network. http://moat.nlanr.net/

18. Ogino, N., Arahata, N.: A decentralized optical bursts routing based on adaptive load splitting into pre-calculated multiple paths. IEICE Transactions on Communications **E88-B**(12), 4507–4516 (2005)

19. Pioro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Morgan Kaufmann (2004)

20. Qiao, C., Yoo, M.: Optical burst switching (OBS) – a new paradigm for an optical internet. J. of High Speed Networks **8**(1), 69–84 (1999)

21. Rosberg, Z., Vu, H., Zukerman, M., White, J.: Performance analyses of optical burst switching networks. IEEE JSAC **21**(7), 1187–1197 (2003)

22. Teng, J., Rouskas, G. N.: Traffic engineering approach to path selection in optical burst switching networks. Journal of Optical Networking **4**(11), 759–777 (2005)

23. Yang, L., Rouskas, G.: Adaptive path selection in optical burst switched networks. IEEE/OSA Journal of Lightwave Technology **24**(8), 3002–3011 (2006)

24. Yang, L., Rouskas, G.: A framework for absolute QoS guarantees in optical burst switched networks. In: Proceedings of IEEE Broadnets 2006. San Jose, CA (USA) (2006)

25. Zalesky, A., Vu, H., Rosberg, Z., Wong, E., Zukerman, M.: Modelling and performance evaluation of optical burst switched networks with deflection routing and wavelength reservation. In: Proceedings of IEEE INFOCOM 2004. Hong Kong, China (2004)

26. Zhang, J., Wang, S., Zhu, K., Datta, D., Kim, Y. C., Mukherjee, B.: Pre-planned global rerouting for fault management in labeled optical burst-switched wdm networks. In: Proceedings of Global Telecommunications Conference (GLOBECOM 2004). Dallas, TX (USA) (2004)

Chapter 7
Problems in Dynamic Bandwidth Allocation in Connection Oriented Networks

Xavier Hesselbach, Christos Kolias, Ramón Fabregat, Mónica Huerta, and Yezid Donoso

Abstract In this chapter, the analysis of the problems emerging from dynamic bandwidth allocation in connection-oriented packet networks is considered in a multilayer scenario, considering IP protocols on top of an MPLS network over an OBS optical network. The issue (and problem) is to maintain and ensure the end-to-end in-sequence routing of packets, combining load balancing in packet switching architectures and bandwidth/flow allocation in MPLS-based architectures to establish the ordering of packets. If load balancing can be achieved by switches or routers, this can greatly facilitate applying load balancing across the network. Traffic characteristics such as QoS (delay bounds, throughput) and burstiness are considered.

Key words: multi-path, quality of service, class differentiation, load balancing, traffic partitioning, MPLS, OBS

Xavier Hesselbach

Deparment of Telematics Engineering, Universitat Politècnica de Catalunya, C/Jordi Girona, 1 i 3, Mòdul C3 – Campus Nord, 08034 Barcelona, Spain, e-mail: `xavierh@entel.upc.edu`

Christos Kolias

Covad Communications, 110 Rio Robles, San Jose, CA 95134, USA, e-mail: `ckolias@gmail.com`

Ramón Fabregat

Institute of Informatics and Applications, University of Girona, Campus Montilivi, 17071 Girona, Spain, e-mail: `ramon@silver.udg.edu`

Mónica Huerta

Electronics and Circuits Department, University Simón Bolívar, Caracas, Venezuela, e-mail: `mhuerta@usb.ve`

Yezid Donoso

Computer Science and Engineering Department, University of Los Andes, Bogotá, Colombia, e-mail: `ydonoso@uninorte.edu.co`

7.1 Introduction

This chapter deals with the description and analysis of the problems emerging from dynamic bandwidth allocation in connection-oriented packet networks in a multi-layer scenario, considering the Internet Protocol (IP) on top of an MPLS (Multi-protocol Label Switching) network and using an optical burst switching (OBS) network by means of a control plane. The bandwidth optimization problem is split into multi-path routing, load balancing, and traffic control.

The issue and problem is to maintain and ensure the end-to-end in-sequence routing of packets, combining load balancing in packet switching architectures and bandwidth/flow allocation in MPLS-based architectures to establish and guarantee the ordering of packets in a packet-switched network. Other relevant issues such as buffer sizing, packet classification, scalability, and redundancy (i.e., in the form of parallel switch architectures) are also of importance. If load balancing can be achieved by switches or routers, this can greatly facilitate implementing load balancing across the network (e.g., over different paths).

With respect to the issue of bandwidth allocation and load balancing in an MPLS network, we will examine a number of different approaches and alternatives for performing load balancing optimization via assigning capacity and carrying out flow allocation in a multi-path MPLS network. Traffic characteristics such as QoS metrics, e.g., delay bounds, throughput, jitter, packet loss, and burstiness (measured as the ratio of the peak data rate over the average data rate), will be considered for optimizing load balancing. In fact, it is determined that our solutions can be implemented in any MPLS-type (e.g., flow granularity) network architecture given the solutions' flexibility and scalability aspects. They can also be extended to multicast network models and models with traffic priorities.

This chapter is organized into three parts:

1. Technological perspective and challenges. We present an overview of the connection-oriented technologies, MPLS networks, load balancing strategies, and multi-path in packet networks.
2. Load Balancing strategies in a multi-path scenario.
3. Intelligent bandwidth allocation algorithms for multilayer traffic mapping with priority provision.

7.2 Technological Perspective and Challenges

This section presents a survey of basic knowledge in technologies, protocols, and algorithms for the next generation of networks. This is a background for the subsequent sections, and exhibits actual network deployment environments where our results can be applied.

7.2.1 Definitions and Concepts Overview

Quality of Service (QoS) refers to the capability of a network to provide service differentiation for certain types of traffic. The primary goal of QoS is to provide specific traffic guarantees: controlled delay and jitter, bandwidth allocation, and packet loss characteristics. Also important is making sure that providing priority to one type of traffic does not make other traffic fail. Fundamentally, QoS can provide for better service to certain traffic by presenting network resource guarantees that satisfy the user's demand. A key feature of QoS is the service level, which refers to the actual end-to-end QoS capability, meaning the capability of a network to deliver service needed by specific network traffic from an edge node (i.e., router) to another edge node. The services differ in their level of QoS "strictness", which describes how tightly the service can be bound by specific bandwidth, delay, jitter, and loss characteristics. Three basic levels of end-to-end (e2e) QoS can be provided across a heterogeneous network:

- Best-effort service: also known as lack of QoS, characterized by FIFO queues which have no traffic differentiation.
- Differentiated service (also called soft QoS): some traffic is treated better than the rest.
- Guaranteed service (also called hard QoS): an absolute reservation of network resources for specific traffic; it is provided through bandwidth reservation mechanisms and the use of CBWFQ (Class-Based Weighted Fair Queueing), an extension of WFQ that can use the EXP field in the MPLS shim header format of the packets as a criterion to allocate a different amount of resource for each CoS (class of service).

It is clear today that IP is the dominant protocol for the vast majority of Internet applications, including those that are characterized as bandwidth-demanding and delay-sensitive (although IP was not originally intended and certainly was not designed to support such applications). An example of an emerging Internet application is video streaming, in particular real-time or live video. Each single stream can consume 1 Mbit/s or so, so the aggregation of video streams uploaded or downloaded by millions of users can certainly overload a traditional network.

Therefore, the new generation of networks should provide a QoS (Quality of Service) degree according to the needs of each user and enough bandwidth. The key problem is how the traffic can be marked to belong to a certain class of traffic or priority and, second and more important, how this huge aggregation of traffic streams can be supported by the network. A solution for this type of problem is the well-known differentiated services (DiffServ) framework.

Over the last few years there has been significant activity in the field of multi-protocol label switching (MPLS), which has emerged as a new networking paradigm that facilitates routing and can implement a form of QoS. In fact, an increasing number of networks already implement MPLS.

MPLS provides a flexible routing mechanism based on the assignment of packets with the same characteristics to complete end-to-end paths within an Autonomous

Domain. "Traffic engineering" is one of the primary goals achieved by MPLS. Furthermore, MPLS can allow for the aggregation of traffic streams, if needed.

MPLS has become one of the primary technologies for supporting traffic engineering on the Internet [10, 12]. An MPLS network is composed of MPLS routers: LSRs (Label Switched Routers) that represent the core of the network (backbone) and LERs (Label Edge Routers) that are the interface between the MPLS domain and other networks.
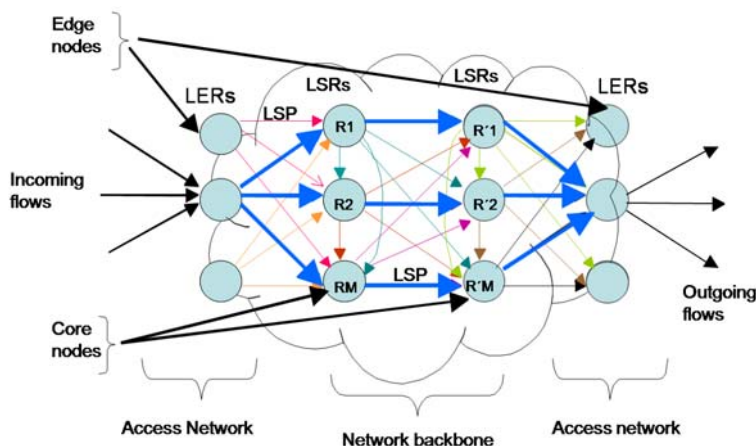

Fig. 7.1 Core and edge nodes in an MPLS network. A distributed LSP

An MPLS network is connection-oriented, where each connection or LSP (Label Switching Path) is established between two LERs, the Ingress and the Egress LER, as depicted in Figure 7.1.

In an MPLS network, multiple paths can be used to forward packets belonging to the same "forwarding equivalent class" (FEC) by explicit routing. Each time an LSP is established, all the LSRs that belong to it must use a label in order to identify the LSP transiting by it, and consequently every packet of this LSP must carry this label encoded inside it when arriving at that LSR. When a packet is received by an LSR, the LSR must look for the packet label and search for a Next Hop Label Forwarding Entry (NHLFE) that refers to this label in order to decide which interface will be used to reach the next hop in the network.

Our interest focuses on the QoS operation for a single node perspective, but we will extend this to a sequence of nodes, and so, to a complete network. One way network elements handle an overflow of arriving traffic is by using a certain queuing algorithm to sort the traffic and then determining some method of routing the traffic appropriately (i.e., via prioritization) onto an outgoing link.

7.2.2 Node Model for Packet Networks with Traffic Prioritization

There are a number of queuing tools, each one designed to solve a specific network traffic problem and having a particular effect on network performance. In Figure 7.2 we show a network node model and its main parameters. In this model, *CoS* (class of service) is defined as a traffic classification, according to a set of degrees of quality (mainly for bandwidth, delay, and packet losses).

The parameters used to model a node are as follows:

- λ_{IN} (bit/seg) represents all incoming traffic that arrives at the router. This aggregated traffic is assumed to be Poisson.
- λ_i (bit/seg), $i = 1 \ldots, N$, represents the aggregated traffic of CoS_i carried by all the LSPs which are forwarded to the current output interface. Note that this traffic is not Poisson (as the input queue is finite), but in our analysis we have approximated it as if it were under the hypothesis that losses in the input queue are very low.
- $\mu_{PROCESS}$ (bit/seg) models the service rate at which packets are classified.
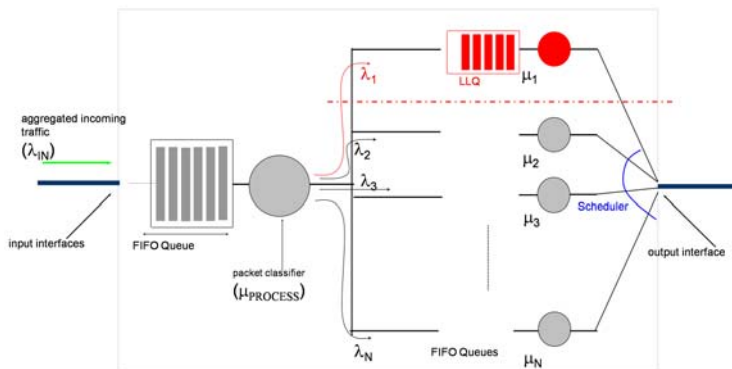- μ_i (bit/seg), $i = 1, \ldots, N$, represents the service class rate.

Fig. 7.2 Node model for a packet network considering prioritized differentiated services

7.2.3 QoS in IP/DiffServ/MPLS and in OBS Networks

In Optical Burst Switching (OBS), several signaling schemes have been proposed to support QoS. The basic problem is to determine the criteria for choosing a signaling method. The selection can be made according to several parameters, such as complexity of the algorithm or optimization of a certain parameter, but what the

user applications need is a QoS with respect to bandwidth, delay bounds, jitter, and losses. This is what an OBS network should provide, in a way that can be managed by the upper layers, to allocate classes of service according to existing standards for MPLS, DiffServ, and IP traffic.

The main scheme for controlling (or administrating) the QoS in OBS is the control of the offset (the time delay introduced at the edge of the network where the burst will be transmitted, to set up the hardware of the next node to forward the optical burst), in order to isolate different classes of bursts, such as high- and low-priority bursts. In this kind of method, extra offset is given to the higher priority bursts, to reserve resources further in advance of the arrival of the burst, increasing the probability of a successful reservation. For offset schemes, there are currently discussions being carried out with respect to prioritization as part of the contention resolution scheme, and also for an absolute QoS provision [14].

7.2.4 Optical Burst Switching Using MPLS

Semipermanent data pipes can be set up between different ingress-egress router pairs using an MPLS-type technique. MPLS uses labels to make forwarding decisions at the network nodes LSR, in contrast to the traditional destination-based hop-by-hop forwarding in IP networks. In MPLS, the space of all possible forwarding options is partitioned into Forwarding Equivalence Classes (FECs). For example, all the packets destined for a given egress and having the same quality of service (QoS) may belong to the same FEC.

The packets are labeled at the ingress depending on the FEC to which they belong. Each of the intermediate nodes, the LSRs, uses the label of the incoming packet to determine its next hop, and also performs label swapping (i.e., replaces the incoming label with the new outgoing label that identifies the respective FEC for the downstream node). Such a label-based forwarding technique reduces the processing overhead required for routing at the LSRs, thereby improving their packet forwarding performance and scalability. Also, the label swapping process used in MPLS creates multi-point-to-point packet forwarding trees, in contrast to a routing mesh in conventional networks.

MPLS can play a major role in traffic engineering and improving the throughput performance of an OBS-based network, as described below. Each cross-connect in the optical backbone will have label swapping information about the precomputed routes in its label information base (LIB).

An LIB can be set up using standard techniques such as routing protocols with traffic engineering extensions to distribute information about the optical domain (available bandwidth per wavelength, number of wavelengths per fiber) and Constraint-Based Routing Label Distribution Protocol (CR-LDP) or Resource Reservation Protocol (RSVP) to distribute labels. Whenever an ingress router has a data burst to transmit, it refers to its LIB, to determine the appropriate label. This label is

included in the control packet that precedes this data burst. When the control packet arrives at any of the intermediate nodes, the following actions take place:

- The label in the control packet is used to point to the data burst forwarding information in the LIB, such as the output interface and any priority or QoS information.
- The cross-connect is set up to switch the data burst corresponding to that control packet in the all-optical domain. For this, information in the control packet about the length and offset of the data burst is used in addition to the forwarding information derived from the LIB. In particular, the latter is used to determine the mapping from the incoming fiber and wavelength to the outgoing fiber and wavelength. In order to be able to forward successive data bursts of the same connection (LSP) on different wavelengths in a given fiber, we propose that the label only specify incoming-fiber-to-outgoing-fiber mapping, while the information about the wavelength be appended to the outgoing label at every hop. The LIB may furnish other QoS information as well. Examples include defining a subset of candidate wavelengths on the outgoing fiber, determining the eligibility of that data burst to use wavelength conversion, stating whether (in case of contention) the control packet is allowed to preempt some reservation already acquired by the control packet of low-priority data burst, and so on.
- The control packet then undergoes label swapping (and wavelength information appending) and is forwarded on the dedicated control channel of the outgoing fiber as indicated by the LIB.

7.3 Load Balancing Strategies in a Multi-path Scenario

Load Balancing is a key mechanism in traffic engineering. The strategy of multi-path routing with load balancing enhances the network throughput. The use of effective preordering packet functions optimizes network utilization and reduces packet disordering and imbalance. This section presents a model to study the impact of packet preordering in multi-path MPLS networks, and the traffic partitioning to implement a flow partitioning based on an optimization model. Some experimental results from an optimized network are presented. A multi-objective traffic engineering scheme (GMM, or Generalized Multicast Multi-path, model) [3] using different distribution trees to multicast several flows has been proposed. Solving the GMM model allows us to compute the flow components required at the ingress LER mapped to the set of egress nodes assigned to each link. We present an effective hashing strategy to handle traffic partitioning from the GMM model. The GMM model considers a network represented as a graph $G(N, E)$, with N denoting the set of nodes and E the set of links. The cardinality of a set is denoted as $|.|$; thus $|N|$ represents the cardinality of N. The set of flows (or commodities) is denoted as F.

Each flow $f \in F$ can be split into $|K_f|$ subflows that after normalization can be denoted as f_k, $k = 1, \ldots, |K_f|$. In this case, f_k indicates the fraction of $f \in F$ it

transports, $\sum_{k=1}^{|K_f|} f_k = 1$. For each flow $f \in F$ we have a source $s_f \in N$ and a set of destination or egress nodes $T_f \subset N$. Let t be an egress node, i.e., $t \in T_f$. Let $X_{ij}^{f_k t}$ denote the fraction of subflow f_k to egress node t assigned to link $(i,j) \in E$, i.e., $0 \leq X_{ij}^{f_k t} \leq 1$. In this way, the n components of decision vector X are given by all $X_{ij}^{f_k t}$. Note that $X_{ij}^{f_k t}$ uses five indices: i, j, f, k, and t. The novel introduction of a subflow index k gives an easy way to identify subflows and define LSPs in an MPLS implementation. Let c_{ij} be the capacity (in bit/s) of each link $(i,j) \in E$. Let b_f be the traffic request (measured in bit/s) of flow $f \in F$, traveling from source s_f to T_f. Let d_{ij} be the delay (in ms) of each link $(i,j) \in E$. The binary variables $Y_{ij}^{f_k t}$ represent whether a link (i,j) is being used (value 1) or not (value 0) for transporting subflow f_k to destination node t,

$$Y_{ij}^{f_k t} = \left\lceil X_{ij}^{f_k t} \right\rceil = \begin{cases} 0; & \text{if } X_{ij}^{f_k t} = 0 \\ 1; & \text{otherwise} \end{cases} \tag{7.1}$$

where $\lceil . \rceil$ denotes the ceiling function. Finally, let $connection_{ij}$ be an indicator of whether there is a link between nodes i and j. Given the above notation, the proposed GMM model considers the following objective functions:

- Maximum link utilization:

$$\max_{(i,j) \in E} \alpha_{ij} \tag{7.2}$$

where

$$\alpha_{ij} = \frac{1}{c_{ij}} \sum_{f=1}^{|F|} \sum_{k=1}^{|K_f|} b_f \left\{ \max_{t \in T_f} X_{ij}^{f_k t} \right\} \tag{7.3}$$

- Hop count, in several different ways, such as total hop count,

$$\sum_{(i,j) \in E} \sum_{f \in F} \sum_{k \in K_f} \sum_{t \in T_f} Y_{ij}^{f_k t}, \tag{7.4}$$

average hop count,

$$\frac{\sum_{(i,j) \in E} \sum_{f \in F} \sum_{k \in K_f} \sum_{t \in T_f} Y_{ij}^{f_k t}}{\sum_{f \in F} \sum_{k=1}^{|K_f|} |T_f|}, \tag{7.5}$$

a maximum hop count, which is useful for QoS assurance,

$$\max_{f \in F} \max_{k \in K_f} \max_{t \in T_f} \sum_{(i,j) \in E} Y_{ij}^{f_k t}, \tag{7.6}$$

or a maximum hop count variation for a flow, which is useful for jitter and queue size calculations,

$$\max_{f \in F} \max_{t \in T_f} H_{ft} \tag{7.7}$$

where

$$H_{ft} = \max_{k \in K_f} \left\{ \sum_{(i,j) \in E} Y_{ij}^{f_k t} \right\} - \min_{k \in K_f} \left\{ \sum_{(i,j) \in E} Y_{ij}^{f_k t} \right\} \tag{7.8}$$

- Delay measures, such as total delay,

$$\sum_{(i,j) \in E} \sum_{f \in F} \sum_{k \in K_f} \sum_{t \in T_f} d_{ij} \cdot Y_{ij}^{f_k t}, \tag{7.9}$$

- average delay,

$$\frac{\displaystyle\sum_{(i,j) \in E} \sum_{f \in F} \sum_{k \in K_f} \sum_{t \in T_f} d_{ij} \cdot Y_{ij}^{f_k t}}{\displaystyle\sum_{f \in F} \sum_{k=1}^{|K_f|} |T_f|}, \tag{7.10}$$

a maximum delay, which is useful for QoS assurance,

$$\max_{f \in F} \max_{k \in K_f} \max_{t \in T_f} \sum_{(i,j) \in E} d_{ij} Y_{ij}^{f_k t}, \tag{7.11}$$

and a maximum delay variation for a flow, which is useful for jitter and queue size calculations,

$$\max_{f \in F} \max_{t \in T_f} \Delta_{ft} \tag{7.12}$$

where

$$\Delta_{ft} = \max_{k \in K_f} \left\{ \sum_{(i,j) \in E} d_{ij} \cdot Y_{ij}^{f_k t} \right\} - \min_{k \in K_f} \left\{ \sum_{(i,j) \in E} d_{ij} \cdot Y_{ij}^{f_k t} \right\}. \tag{7.13}$$

- Total bandwidth consumption:

$$\sum_{(i,j) \in E} \sum_{f \in F} \sum_{k \in K_f} b_f \left\{ \max_{t \in T_f} X_{ij}^{f_k t} \right\} \tag{7.14}$$

A MOP (multi-objective problem) formulation usually considers several constraints such as

- *Flow conservation* constraints:

$$\sum_{j=1}^{|N|} \sum_{k=1}^{|K_f|} X_{s_f j}^{f_k t} \qquad\qquad = 1 \qquad \forall f \in F \tag{7.15a}$$

$$\sum_{i=1}^{|N|} \sum_{k=1}^{|K_f|} X_{it}^{f_k t} \qquad\qquad = 1 \qquad \forall f \in F, t \in T_f \tag{7.15b}$$

$$\sum_{j=1}^{|N|} X_{i_f j}^{f_k t} - \sum_{i=1}^{|N|} X_{i_f i}^{f_k t} \qquad = 0 \qquad \forall f \in F, k \in K_f, t \in T_f, i_f \in N \setminus (\{s_f\} \cup T_f) \tag{7.15c}$$

- *Subflow uniformity* constraints to ensure that a subflow f_k always transports the same information:

$$X_{i'j'}^{f_k t'} - X_{ij}^{f_k t} \quad \leq 1 - Y_{ij}^{f_k t} \quad \forall f \in F, k \in K_f, (i,j), (i',j') \in E, t, t' \in T_f \tag{7.16a}$$

$$X_{ij}^{f_k t} \qquad\qquad \leq Y_{ij}^{f_k t} \qquad \forall f \in F, k \in K_f, (i,j) \in E, t \in T_f \tag{7.16b}$$

Without this restriction, $X_{ij}^{f_k t} > 0$ may differ from $X_{i'j'}^{f_k t'} > 0$. Therefore, the same subflow f_k may not transport the same data to different destinations t and t'. As a consequence of this new constraint, mapping subflows to LSPs becomes easy.

- *Link capacity* constraints:

$$\sum_{f=1}^{|F|} \sum_{k=1}^{|K_f|} b_f \left\{ \max_{t \in T_f} X_{ij}^{f_k t} \right\} \qquad\qquad \leq c_{ij} \qquad\qquad \forall i, j \in N \tag{7.17}$$

- Constraints on the maximum number of subflows:

$$\sum_{k=1}^{|K_f|} \sum_{j \in N} Y_{ij}^{f_k t} \qquad\qquad \leq N_{\max} \qquad\qquad \forall f \in F, t \in T_f, i \in N, \tag{7.18}$$

or alternatively, depending on required bandwidth b_f:

$$\sum_{k=1}^{|K_f|} \sum_{j \in N} Y_{ij}^{f_k t} \quad \leq b_f \frac{\sum_{j \in N} connection_{ij}}{\sum_{j \in N} c_{ij}} \qquad \forall f \in F, t \in T_f, i \in N \tag{7.19}$$

In summary, the proposed GMM model follows the general mathematical framework of any MOP. The model considers 11 objective functions and seven classes of constraints [3]. Clearly, it is not difficult to increment the number of objectives or constraints of the proposed model if new ones appear in the literature or they are useful for a given situation. In fact, Packet Loss was not considered in this proposal, but including it would be very easy. Anyway, this model is very useful for traffic balancing, because it can provide an optimized path for the flows of data. So, let us have a look at the Load Balancing strategy in practice.

7.3.1 Load Balancing in Packet Networks in Multicast Multi-path Scenarios

Load Balancing [11] provides an extremely useful mechanism for implementing traffic engineering. The aim of load balancing is routing traffic using links that are less congested according to well-known criteria. This can result in small delays compared to a default flow routing. For a load balancing model to be general, unicast considerations are not enough and multicast should also be considered. One interesting solution to the balancing alternative is the multi-path approach, in which data is transmitted through different paths inside a network. One of the most difficult issues in load balancing is how to guarantee the end-to-end delay among several Label Switched Paths (LSPs), while maintaining packet sequencing. This requirement is especially important for the network throughput for the TCP protocol, because packet disordering can produce false congestion detections.

The load balancing techniques can be classified in two groups, one based on the (logical) connection established, which is represented by a small number of parameters and where routing decisions affect the whole flow, and another based on the packets, where decisions are made on a per packet basis and which is therefore simpler than the first one; the latter is the technique considered here.

When an IP packet ingresses into the MPLS domain, the ingress Label Edge Router (LER) analyzes the header. Depending on information of the destiny, type of traffic, etc., an MPLS label is assigned. It consists of a short, fixed-length identifier (20 bits) associated with the path that the packet will have to take into account in order to reach the egress node.

According to the scale, the traffic engineering (TE) mechanisms are classified into two basic types:

- Time Dependent, where the traffic control algorithms optimize the network resources in response to traffic variations in a long time range.
- State Dependent, where the traffic control algorithms respond immediately to state variations in the network. In other words, it adapts changes to a short time range.

There are many subproblems involved in the performance optimization of operational MPLS networks. Three of the most significant problems include

1. Constraint-based routing.
2. Traffic partitioning and assignment.
3. Restoration.

It should be noted that even though these problems are well known in other application domains, they are still in a state of infancy with respect to MPLS, and much remains to be done.

Another important capability MPLS provides is constraint-based routing. The ingress node, the Label Edge Router (LER), establishes an explicit route through the network. Rather than inefficiently carrying the explicit route in each packet, MPLS allows the explicit route to be carried only at the time the label switched path

(LSP) is set up. The subsequent packets traversing this path are forwarded using packet labels. Constraint-based routing is potentially useful for traffic engineering.

In the current literature these areas are addressed in a somewhat limited way. Constraint-based routing deals, in general, with the computation of paths for LSPs subject to various types of constraints. The constraints themselves may be inherent in the network (e.g., available bandwidth) or they can be administratively specified (e.g., affinities and resource class attributes, and diversity requirements for protection and restoration). D-LSP (Distributed LSP) is constructed by partitioning an LSP into several sub-LSPs assigned over different nodes that belong to "disjoint routes." The arriving traffic streams are allocated to the sub-LSP at the ingress LER. We do not necessarily assume that each disjoint node route has the same number of hops. A network model can be defined, and the architecture of a D-LSP is represented with an MPLS network model. Figure 7.3 shows an example of a D-LSP established in the MPLS network model.

The D-LSP is originated from an ingress LER and is destined for an egress LER. A D-LSP is partitioned into sub-LSPs and spread over the disjoint node routes. The load balancing techniques can take advantage of this scheme. Core LSRs provide transit services inside the network, while LERs provide an interface with the external networks. An ingress LER may assign one or more paths to a given egress to an MPLS domain, and using these LSPs the traffic load can be balanced across a complex topology.

The capability to divide the traffic offers several advantages, one of the most important being the distribution of the flow and the optimization of the bandwidth.

In this research work a multicast approach is proposed. Multicast connections are connections between one or more senders and a number of members of a group. The aim of multicasting is to be able to concurrently send data from a (single) sender to the (multiple) members of a group in an efficient manner. In this case, instead of creating multiple paths to transmit a traffic flow from the ingress node to just one egress node it is necessary to create multiple trees to transport the flow from the ingress node to the egress node set of the multicast group.

Many multicast applications, such as audio- and videoconferencing, or even collaborative environments and distributed interactive simulations, have multiple quality-of-service requirements in relation to bandwidth, packet delay, packet loss, cost, and so on. In multicast transmission, load balancing consists of traffic being split (using the multi-path approach), across multiple trees, between the ingress node and the set of egress nodes. In multicasting (host) nodes can enter or leave the transmission tree as they wish, which makes the connections rather dynamic.

In MPLS, unicast and multicast packets have already been assigned a different type as indicated by the IPv4 unicast or multicast address. Therefore, MPLS routers know whether a packet belongs to a unicast or a multicast flow. In the case of unicast forwarding the event of an incoming flow leads to the forwarding of exactly one flow. The packet duplication mechanism that is implemented in IP routers to support the IP multicasting can be used to duplicate MPLS packets

MPLS routers at the bifurcation of a multicasting routing tree duplicate packets and send copies of the same packet on different outgoing links. In this case, an
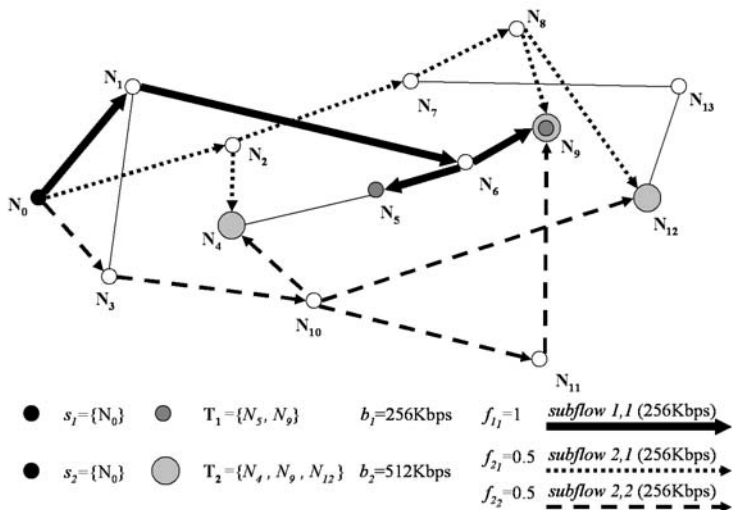
Fig. 7.3 Example of subflow (partition of a flow) representation in an NSF network

MPLS label is pushed into the multicast packet according to next hop branching node router. Upon arriving at a next hop branching node router, the label is pulled out and again the same process is repeated. This process should be repeated until the packet reaches its destination.

7.3.2 Model for Traffic Partitioning

A load balancing system is in general formed by a Traffic Splitter and several outgoing links (up to M_{max} in Figure 7.4).
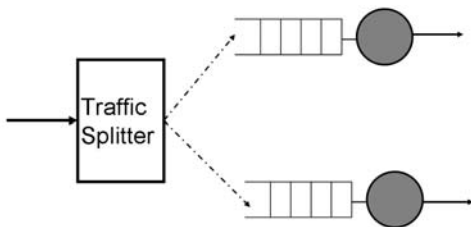

Fig. 7.4 Model of a traffic splitter

Conceptually, the input traffic is partitioned according to certain criteria into M_{max} bins at the MPLS ingress node. The M_{max} bins are mapped onto the pre-

established LSPs. The fraction assigned to each LSP is calculated using an opti-
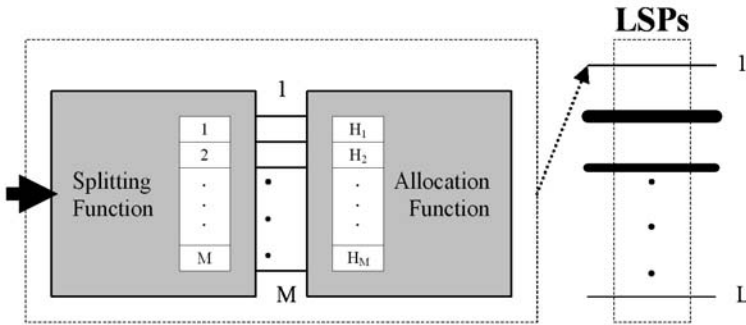mization model.

Fig. 7.5 Functional model for a multi-path load balancing

A load balancing mechanism is functionally comprised of a Splitting Function
and an Allocation Function (Figure 7.5). The Splitting Function, partitions and allo-
cates incoming traffic to the outgoing links, guaranteeing the order of packets. The
Allocation Function determines the LSP where every packet have to be forwarded
to the egress router and the time it must be delivered.

In [3] a multi-objective traffic engineering scheme (GMM model) using different
distribution trees to multicast several flows has been proposed. Solving the GMM
model allows us to compute the fraction of flow demanded from the ingress node to
the set of egress nodes assigned to each link. We propose effective hashing strategies
to handle traffic partitioning from the GMM model. The GMM model considers a
network represented as a graph $G(N,E)$, with N denoting the set of nodes and E the
set of links. The set of flows is denoted as F. Each flow $f \in F$ can be split into K_f
subflows. It can be denoted (following normalization) as f_k, $k = 1, \ldots, K_f$, which
indicates the fraction of f transported. We have found that employing Table-based
Hashing provides better performance compared to Direct Hashing, due to the load
distribution because of the unequal weights. Hashing values are tuned according to
the x_{ij}^{fkt} parameters calculated from the network optimization.

Hashing-based traffic partitioning algorithms are simple to compute and inde-
pendent of the state of the network. A good hash function satisfies the assumption
of simple uniform hashing, that is, each key is equally likely to hash to any of the
L outgoing links, independently of where any other key has hashed to. In practice,
heuristic techniques are frequently used to define a hash function that works well.
Hashing schemes for load balancing can be classified into Direct Hashing and Table-
based Hashing. The L value is the number of different paths that can be established
from the source ingress router to a certain egress router. Therefore, L depends on
the topology of the network. On the other hand, the value of $M_{\max}$ can be tuned
according to the allocation function and the Load Balancing Mechanisms.

In order to dimension the egress buffers, we turn our attention to the following scheme, which requires a packet reordering algorithm at the egress node to prevent packets from getting out of sequence and to also make egress buffering more efficient. Taking into account that faster paths require buffer allocation in order to synchronize packets received from slower paths, we need to calculate the buffer size required. We define the following notation.

Consider a single flow f. For an f_k subflow from this flow f to an egress node t, the end-to-end delay is

$$d^{f_k t} = \sum_{(i,j) \in E} d_{ij} \cdot Y_{ij}^{f_k t} \tag{7.20}$$

where d_{ij} is the delay (in milliseconds) of each (i, j) link in the network.

The delay for the slowest f_k belonging to the flow f to egress node t is

$$d_{\text{slowest}}^{f_k t} = \max_{\forall k \in K_f} \{d^{f_k t}\}. \tag{7.21}$$

Let $f_k t_{\text{slowest}}$ be the f_k with a $d_{\text{slowest}}^{f_k t}$ delay. Then buffer size $B^{f_k t}$ required for each f_k flow is

$$B^{f_k t} = \left(d_{\text{slowest}}^{f_k t} - d^{f_k t}\right) \cdot \left(b_f \cdot X_{\text{closest\_link\_to\_node\_t}}^{f_k t}\right) \tag{7.22}$$

where $b_f \cdot X_{\text{closest\_link\_to\_node\_t}}^{f_k t}$ is the bit rate arriving to node t from flow f_k. Note that a buffer for the slowest path is not required.

Now, the total buffer size in an egress node t for a single flow f_k is

$$B^{ft} = \sum_{k=1}^{|K_f|} B^{f_k t}. \tag{7.23}$$

7.4 Intelligent Bandwidth Allocation Algorithms for Multilayer Traffic Mapping with Priority Provision

New OBS algorithms enable the transmission of a single optical burst at the physical layer of the network. Requirements in upper layers suggest the need of a control plane and an efficient mapping procedure for packet encapsulation in order to organize the traffic classes according to statistical behavior and available paths.

One important issue is the design of suitable mapping algorithms for QoS guarantees. The significance of the offset time in the control packets is studied in order to optimize the overall throughput of the network. The management of the offset time leads to interesting problems of burst selection criteria and scheduling.

7.4.1 QoS Algorithms for OBS

A number of approaches for QoS provisioning in OBS networks have been proposed in the literature. These approaches can be classified into offset-based, strict priority, segmentation-based, and proportional QoS. The main aim of these proposals is to provide relative service differentiation with regard to packet loss probability. The use of classical fair scheduling algorithms in the data plane of optical nodes has generally been avoided in the literature. This is due to the absence of the concept of "packet queues" in optical nodes, beyond the number of packets that can be buffered (while in-flight) in Fiber Delay Lines.

Packets arriving at the ingress node are classified into one of the FECs already defined by the network administrator. These packets are then shaped and a policy is applied to conform to the QoS requirements of their FEC. Packets are then assembled into bursts in the burst assembly queue. The burst assembly process is carried out using the containerization with aggregation-timeout (CAT) algorithm. In principle, the function of the CAT algorithm is to assemble the arriving packets into data bursts, such that each data burst contains packets that belong to one FEC. Two parameters control this burst assembly process in CAT, namely the maximum burst size (B_{max}) and burst assembly timeout (T_{max}). The maximum burst size controls the maximum number of packet bytes contained in a single burst. If the incoming traffic intensity is high enough, the maximum burst size will be reached in a relatively short time. If, on the other hand, the traffic has a low arrival rate, the burst being assembled might have to be queued for a relatively long time until the maximum burst size is reached. In order to avoid large queuing delays for bursts, the assembly timeout parameter is used to release the burst under assembly. Thereafter, data bursts are inserted into the burst queue and scheduled for processing by the reservation manager according to their QoS requirements.

We propose the use of Fair Packet Queueing (FPQ) algorithms for scheduling the processing of data bursts by the reservation manager. The FPQ scheduling disciplines have three desirable properties:

1. they can guarantee an upper bound on delay to a token bucket-constrained session,
2. they guarantee the upper bound on delay regardless of the behavior of other sessions (isolation)
3. they can ensure relative fairness in bandwidth allocation among backlogged sessions.

The particular choice of the scheduler is not imposed by the architecture. There exist a number of FPQ algorithms in the literature. For example, Weighted Fair Queueing (WFQ), Self-Clocked Fair Queueing (SCFQ), and Start-time Fair Queueing (SFQ) [13].

The FPQ scheduler selects from the bursts queue the eligible data burst to be processed by the wavelength reservation manager. There is a significant difference between the proposed usage of the FPQ scheduler and its usage as a conventional packet scheduler. In the latter case, the packet selected by the FPQ scheduler is

directly sent for transmission, whereas in our architecture the FPQ scheduler regulates the access to the reservation manager. This difference appears more clearly when calculating the queuing delay in the burst queue, since the processing delay in the reservation manager is independent of the burst's length.

Considering that the modes currently under investigation for the implementation of the next generation coarse packet-switched optical networks, e.g., Burst Switching, are based on these reservation policies, we briefly present the resource reservation protocols for OBS.

Initially, two main protocols were used in OBS: Tell-And-Wait (TAW) and Tell-And-Go (TAG). The Just-In-Time (JIT) and Just-Enough-Time (JET) protocols were later developed.

In TAG, the burst is emitted by an ingress node even if the establishment of an optical virtual path has not been completed. The burst follows the virtual path in parallel with the setup phase. The data burst and the setup message are spaced by a guard time; this time allows the optical nodes to be set before the burst arrives. When a burst arrives at the egress node, an acknowledgment is sent back to the source after a round-trip delay, and the burst is sent out. If the request bandwidth cannot be granted at an intermediate node, the burst is not lost, and will be transmitted after a backoff time. This protocol does not allow QoS or service differentiation.

In TAW the burst is emitted by an ingress node only if an optical virtual path has been set up through the network to the egress node, by sending a short request message and "waiting" for the acknowledge (ACK) message. When an intermediate node receives this message it makes a reservation using an available wavelength for the requested output; if the requested bandwidth is successfully reserved on all the links along the path, the ACK is sent back to the source after the round-trip delay, and the burst is send out immediately; otherwise, a negative acknowledgment (NACK) will return to release the previously reserved bandwidth, and the source will have to try to make another request after a backoff time. If the request bandwidth cannot be granted at an intermediate node, the resource at the previous nodes will be wasted until the release messages arrive, but the burst is not lost, and will be sent after a backoff time. The resource use is not efficient because reserved links last longer than the burst duration.

In JIT a burst transmission request is sent to a central scheduler. The scheduler then informs each requesting node of the exact time to transmit the data burst; at the appropriate time, the source transmits its burst, and intermediate switches are set as "just in time", for efficient use of wavelength channels, which are set up only for the required burst transmission time. Using computing power and communication between the switches to avoid bandwidth wastage, the central scheduler and the burst assembly allow for providing service differentiation and traffic guarantees over a reserved end-to-end lightpath. Centralized protocols are neither scalable nor robust; a distributed control scheme would be preferred; however, such a scheme relies on synchronization and fast distribution of information on the state of the network, provided by a distributed version of JIT called Reservation with Just-In-Time (RIT), which requires a copy of the request to be sent to all switches (each has a scheduler) concurrently. The problem typically lies with the scheduler implementa-

Table 7.1 OBS signaling protocols for QoS provisioning in IP/DiffServ/MPLS networks

Strategy	Resources usage	End-to-end delay	Resource reservation	DiffServ
TAG	Efficient	High	Not guaranteed	Not supported
TAW	Not efficient	Low	Not guaranteed	Not supported
JIT	Efficient	Medium	Not guaranteed	Supported
RIT	Efficient	Medium	Not guaranteed	Supported
JET	Efficient	High	Near guaranteed	Supported

tion, which not only needs to be synchronized in time, but also needs to share the same global link status information.

In JET there is a setup message sent by an ingress node, where after an additional offset time the burst is transmitted; this offset time takes into account the delays experienced by the setup message within a node. The header carries the data burst's length, destination, and arrival time; this allows for the reservation of the exact resources required for the transmission time of the burst. This is called reserve-a-fixed duration (RFD) scheme, where a node makes advance reservation of the capacity needed at the corresponding output port. JET is the most prevailing distributed protocol for OBS networks today that does not require any kind of optical delays.

An approach that assigns varying additional offset times to different service classes can provide differentiated services in terms of burst loss probability for classes of different priorities [15]. This reduces the loss rate of high-priority traffic at the expense of an increase in the loss rate of lower-priority traffic. The additional offset time becomes part of the end-to-end delay. However, this extra delay does not affect the total end-to-end delay by much, but this problem should be considered with care. Another problem is that a high-priority burst with large offset times will break the resource's free periods into small pieces, and only the burst with smaller size will have higher probability of finding a free wavelength. JET does not use an ACK message for guaranteed the end-to-end lightpath; this causes loss of the bursts that have to be retransmitted to an upper layer.

Table 7.1 summarizes the advantages and disadvantages of each one of the OBS protocols described. It summarizes the general behavior of each signaling protocol according to the requirements of the traffic to be transported. It is evident that JET is a good candidate for those networks where delay is the key constraint, but resource reservations are not guaranteed. Therefore, it can be somewhat less useful for constant bit rate traffic with real time requirements. In this case, JIT or RIT can fulfill these constraints but the end-to-end delay is larger than in JET. We conclude that a new signaling protocol is required taking into account the parameters for control packet management in OBS.

7.5 Conclusions

The chapter surveys several recent open problems in connection-oriented packet networks in the broadband backbone, such as the IP/MPLS/optical technologies. The emerging services requires some QoS degree according to the individual needs, and so several strategies are shown from the point of view of modeling and formulation. The new high-speed optical networks arise in similar problems but different physical technology, all of them under the traffic engineering concepts and the dynamic (auto)provisioning and allocation, some of them shown in this chapter.

References

1. Anguera, D., Hesselbach, X.: Load Balancing algorithms applied to CAC function in MPLS networks. In: Proc. WGN5: V Workshop in G/MPLS networks. Girona, Spain (2006)
2. Chen, J., Chan, S.: Multipath Routing for Video Unicast over Bandwidth-Limited Networks. In: Proc. IEEE GLOBECOM 2001
3. Donoso, Y., Fabregat, R., Fàbrega, L.: Multi-Objective Scheme over Multi-Tree Routing in Multicast MPLS Networks. In: Proc. ACM/IFIP LANC 2003
4. Donoso, Y., Fabregat, R., Marzo, J. L.: Multi-Objective Optimization Algorithm for Multicast Routing with Traffic Engineering. In: Proc. IEEE ICN 2004
5. Hesselbach, X., Fabregat, R., Baran, B., Donoso, Y., Solano, F., Huerta, M.: Hashing based traffic partitioning in a multicast-multipath MPLS network model. In: Proc. IFIP/ACM LANC 2005. Cali, Colombia (2005)
6. Hesselbach, X., Fabregat, R., Kolias, C.: The impact over the packets sequence at the output interface in load balancing strategies. In: Proc. ICTON 2006. Nottingham, United Kingdom (2006)
7. Huerta, M., Hesselbach, X., Fabregat, R.: An Approach to Optimizations Links Utilization in MPLS Networks. In: Proc. ICCGI'06. IEEE. Bucharest, Romania (2006)
8. Izmailov, R., Niculescu, D.: Flow splitting approach for path provisioning and path protection problems. In: Proc. HPSR 2002
9. Kim, C., Choi, Y., Seok, Y., Lee, Y.: A Constrained Multipath Traffic Engineering Scheme for MPLS Networks. In: Proc. IEEE ICC 2002
10. Kim, J., Kim, C., Seok, S., Kang, C.: Traffic Engineering using Adaptive Multipath- Forwarding Against Dynamic Traffic in MPLS Networks. In: Proc. IEEE ICN 2004
11. Long, K., Zhang, Z., Cheng, S.: Load Balancing in MPLS Traffic Engineering, In: Proc. IEEE Workshop on High Performance Switching and Routing. Dallas, USA (2001)
12. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031. January 2001
13. Stephens, D. C., Bennet, J. C. R., Zhang, H.: Implementing scheduling algorithms in high-speed networks, IEEE Journal on Selected Areas in Communications, **17**(6), 1145–1158 (1999)
14. Wu, B., Qin, Y., Xie, F., Feng, G.: Providing absolute QoS in an OBS/WR architecture. Journal of Optical Networking, **7**(9), 796–813 (2008)
15. Yoo, M., Qiao, C., Dixit, S.: Optical burst switching for service differentiation in the next-generation optical internet. IEEE Communications Magazine, **39**(2), 98–104, (2001)

Chapter 8
Optimization of OSPF Routing in IP Networks

Andreas Bley, Bernard Fortz, Eric Gourdin, Kaj Holmberg, Olivier Klopfenstein,
Michał Pióro, Artur Tomaszewski, and Hakan Ümit

Abstract The Internet is a huge world-wide packet switching network comprised of
more than $13,000$ distinct subnetworks, referred to as Autonomous Systems (ASs).
They all rely on the Internet Protocol (IP) for transport of packets across the net-
work. And most of them use shortest path routing protocols, such as OSPF or IS-IS,
to control the routing of IP packets within an AS. The idea of the routing is ex-
tremely simple — every packet is forwarded on IP links along the shortest route
between its source and destination nodes of the AS. The AS network administrator
can manage the routing of packets in the AS by supplying the so-called admin-

Andreas Bley
Institute for Mathematics, Technical University Berlin, Str. des 17. Juni 136, D-10623 Berlin, Ger-
many, e-mail: `bley@math.tu-berlin.de`

Bernard Fortz
Départment d'Informatique, Faculté des Sciences, Université Libre de Bruxelles (ULB), Belgium,
and
CORE, Université catholique de Louvain, Belgium, e-mail: `bernard.fortz@ulb.ac.be`

Eric Gourdin · Olivier Klopfenstein
France Telecom, Orange Labs R&D, France, e-mail: `eric.gourdin,olivier.`
`klopfenstein@orange-ftgroup.com`

Kaj Holmberg
Linköping Institute of Technology, SE-581 83 Linköping, Sweden, e-mail: `kahol@mai.liu.se`

Michał Pióro
Institute of Telecommunications, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-
665 Warsaw, Poland, and
Department of Electrical and Information Technology, Lund University, Box 118, SE-221 00 Lund,
Sweden, e-mail: `mpp@tele.pw.edu.pl`

Artur Tomaszewski
Institute of Telecommunications, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-
665 Warsaw, Poland, e-mail: `artur@tele.pw.edu.pl`

Hakan Ümit
Louvain Management School, Belgium, and
CORE, Université catholique de Louvain, Belgium, e-mail: `hakan.umit@uclouvain.be`

istrative weights of IP links, which specify the link lengths that are used by the routing protocols for their shortest path computations. The main advantage of the shortest path routing policy is its simplicity, allowing for little administrative overhead. From the network engineering perspective, however, shortest path routing can pose problems in achieving satisfactory traffic handling efficiency. As all routing paths depend on the same routing metric, it is not possible to configure the routing paths for the communication demands between different pairs of nodes explicitly or individually; the routing can be controlled only indirectly and only as a whole by modifying the routing metric. Thus, one of the main tasks when planning such networks is to find administrative link weights that induce a globally efficient traffic routing configuration of an AS. It turns out that this task leads to very difficult mathematical optimization problems. In this chapter, we discuss and describe exact integer programming models and solution approaches as well as practically efficient smart heuristics for such shortest path routing problems.

Key words: telecommunication networks, shortest path routing, the Internet, OSPF, ECMP, integer linear programming, heuristics

8.1 Introduction

Traffic routing is a key issue in the design and management of communication networks, including the Internet. The term "routing" has the meaning of forcing the traffic flows to use appropriate, frequently predefined, routes. In practice, traffic flows appear every time two end users need to communicate or an end user requires some content from a distant server. Since the flows appear in such a dynamic way, the routing decisions must essentially be realized by the network itself: if a new connection is required for a traffic flow between an end user at node A and an end user at node B, some network equipment must decide along which route from A to B this connection must be established. This routing decision must be made in a fraction of a second and in such a way that the user is provided satisfactory quality of service and the consumption of network resources is minimized. This is why the services offered by communication networks rely heavily on routing protocols.

A routing protocol is a set of rules and mechanisms implemented in a network in order to achieve proper routing decisions. There are many different standard routing protocols. Some of them are technology agnostic, some are designed for a particular networking technology such as optical fiber transmission networks like WDM, SDH, and SONET and packet networks like GbE (Gigabit Ethernet), ATM and IP. One of the major tasks of the network operations team within a telecom company is to tune and manage parameters of the implemented version of a routing standard in order to maximize the network's traffic performance. These activities of the operations team, known as "Traffic Engineering" (TE in short), received a lot of attention during the last decade, especially in the domain of the Internet [41, 75].

In the present chapter we will focus our attention on the optimization problems related to Internet routing. The Internet is a packet switching network, an interconnection of more than 13,000 different subnetworks. It uses the Internet Protocol (IP) to transport data packets across the network and the TCP and UDP protocols to control the flow of data packets between end users. Each individual subnetwork, also known as an Autonomous System (AS), is managed by a single administrative entity of the telecom operator. Some of these ASs are huge and deployed world-wide, though most of them are very small and involve only a few nodes called "routers." The routing protocols that are implemented inside an AS and decide how to route the traffic from one node of the AS to another are called Interior Gateway Protocols (IGPs). "Classical" IGPs rely on a simple routing paradigm called shortest path routing (SPR in short); every packet is forwarded on IP links along the shortest route between its source and destination nodes of the AS. Although in the late 1990s, in order to somewhat overcome what was felt at that time as a deficiency of the shortest path protocols providing more flexibility in the design of routes, more complex routing paradigms were proposed for the Internet, in particular in the context of the MPLS (Multi-protocol Label Switching) technology [3, 84]; at present, SPR protocols are still widely and successfully used on the Internet. Simulation experiments, practical trials, and field deployments have shown, despite the original feeling that it would be very difficult to control traffic efficiently through shortest path mechanisms, that with the shortest path routing paradigm pretty good network traffic performance can actually be achieved. This good performance is partly due to recent advances in routing optimization methods and their implementation in network planning systems.

The most frequently deployed IGP protocols are OSPF (Open Shortest Path First, see [61]) and IS-IS (Intermediate System to Intermediate System, see [65]). According to these protocols, each individual router in the AS must acquire and maintain a complete and accurate vision of the topology of the AS (this is done by frequent exchange of the routing protocol's messages between routers), and appropriate information on every link of this topology. The link-related information is limited to the administrative weight of the link, which is an integer value assigned by the network administrator (within the bounds defined by the version of the routing protocol). Using the topology and the administrative weights of links, each router is able to compute its shortest path tree covering the topology graph, or, in other words, a shortest path towards every other router in the network. This information is stored locally in the so called forwarding table and is used when forwarding incoming packets to the outgoing links. The forwarding table determines, for each destination router in the AS network, the outgoing link that should be used in order to reach the next node on the shortest path to that particular destination router. The administrative weights are the only means the network administrator can use to influence and control the routing of traffic in the network. This control is realized in an indirect way: although the network administrator is interested in optimizing the paths used by each traffic demand, with the shortest path routing protocols she cannot directly define a path and assign it to a particular flow. She has to set the values of the weights so that the defined path becomes a shortest path, or even the unique shortest

path, between the end nodes of the flow. This task is not so obvious as it might appear, since modifying the values of the link weights involved in defining a path for a particular flow can change the paths that have been defined for other flows. Setting the values of weights in an optimal way is the basic TE problem considered in the present chapter.

Finding such weight values that all the paths in a predefined set are shortest paths or unique shortest paths is known as the inverse shortest paths problem [10, 33]. Several variants of this problems can be considered according to the number of paths that can be used for a demand, and to the way the routers can handle multiple shortest paths. Going one step further, one can consider the joint problem of defining the best possible paths (with an objective that captures a certain global performance of the network) and a set of administrative weights compatible with these paths. Still another set of decisions that can be embedded in a more global process are planning decisions, where the design of the network itself (including network topology, resource capacity, traffic routing, etc.) is a part of the decision process. Certainly, each additional level of decisions involved implies a considerable increase in the complexity of the problem.

In the next sections of this chapter we will survey the most recent results concerning network optimization problems related to shortest path routing. The chapter is organized as follows. In Section 8.2, the main TE problem is formally described, all the relevant notation is provided, and the related work is summarized. In Section 8.3, an integer programming approach to the TE problem is described. The approach is based on interlacing two phases within a branch-and-cut algorithm: the first phase finds a routing pattern consisting of a set of routing paths, while the second phase is devoted to finding a set of weights compatible with the routing paths selected in the first phase. In Section 8.4, we discuss valid inequalities that can strengthen the linear relaxations of the model that is used in the first phase of the integer programming approach, together with separation algorithms for these inequalities. Section 8.5 is a survey of the heuristic methods. In Section 8.6, the numerical results obtained with the two-phase approach and with the heuristics are provided and analyzed. In Section 8.7, a full direct mixed-integer linear programming formulation for the considered TE problem is presented, and a number of problem variants are discussed. Then, in Section 8.8 we present historical and literature notes. Finally, Section 8.9 presents concluding remarks.

8.2 Problem Description

In this introductory section we will explain the basic notions and notations, informally discuss the shortest path routing problems studied throughout this chapter, and summarize the related work.

8.2.1 Basic Notions and Notations

The network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of the set of nodes $\mathcal{V}$ (called also vertices), and the set of directed links $\mathcal{E}$ (called also arcs), where $\mathcal{E} \subseteq \mathcal{V}^{|2|}$. Note that $\mathcal{V}^{|2|}$ denotes the set of all ordered pairs $(s,t) \in \mathcal{V}^2$ such that $s \neq t$, i.e., $\mathcal{V}^{|2|} = \mathcal{V}^2 \setminus \{(v,v) : v \in \mathcal{V}\}$. For each node $v \in \mathcal{V}$, we define the set of links $\delta^+(v)$ outgoing from node v and the set of links $\delta^-(v)$ incoming to node v, i.e., $\delta^+(v) = \{e \in \mathcal{E} : a(e) = v\}$ and $\delta^-(v) = \{e \in \mathcal{E} : b(e) = v\}$, where $a(e) \in \mathcal{V}$ and $b(e) \in \mathcal{V}$ denote the originating and terminating node of link $e \in \mathcal{E}$, respectively.

Each link $e \in \mathcal{E}$ has a capacity denoted by c_e. Furthermore, each link $e \in \mathcal{E}$ is assigned a so-called link metric or (administrative) *weight* denoted by w_e. Depending on the context, the weight may be a constant or a variable. The vector $w = (w_e : e \in \mathcal{E})$ is referred to as the weight system (or the weight vector, or the weight sequence). Typically, each such link weight has to be a positive integer bounded from above (for example, OSPF assumes that $w_e \in \{1, 2, \ldots, K = 2^{16} - 1\}$). For optimization purposes we will also consider continuous weight systems w with $1 \leq w_e \leq K, e \in \mathcal{E}$, assuming they are regular. A weight system w is called *regular* if, for any two paths that have different lengths, these lengths differ by at least 1. Certainly, this regularity condition is satisfied when weights are positive integers. If the weights can possibly assume positive rational values, then multiplying all weights w_e by an appropriately large positive number α will assure the regularity while preserving the path-length relation (new path length will be equal to α times the original length).

The traffic demand volume generated at a source node $s \in \mathcal{V}$ and destined to a target node $t \in \mathcal{V} \setminus \{s\}$ will be denoted by d_{st}. Such demand volumes are expressed in the same units of bandwidth as capacities of links. If there is no traffic demand for a pair of nodes $(s,t) \in \mathcal{V}^{|2|}$ then we simply put $d_{st} = 0$. The total demand volume destined to node $t \in \mathcal{V}$ will be denoted by D_t so that $D_t = \sum_{s \in \mathcal{V} \setminus \{t\}} d_{st}$.

The set of all elementary (i.e., loop-less) paths in the network graph is denoted by $\mathcal{P}$. Each path $p \in \mathcal{P}$ is represented by its set of links so that $p \subseteq \mathcal{E}$. The length of path $p \in \mathcal{P}$ with respect to weight system w will be denoted by $w(p)$, and any shortest path with respect to system w will be referred to as a w-shortest path. Clearly, $w(p) = \sum_{e \in p} w_e$.

Unless stated otherwise, in the sequel we will always assume that link capacities $c = (c_e : e \in \mathcal{E})$ and demand volumes $d = (d_{st} : (s,t) \in \mathcal{V}^{|2|})$ are fixed and given.

8.2.2 Informal Formulation

The basic problem considered in this chapter is called the *shortest path traffic engineering problem* (STEP). Its most commonly known version assumes unique shortest paths and consists of finding a *routing pattern*, i.e., set of paths $\hat{\mathcal{P}} = \{\hat{p}_{st} : (s,t) \in \mathcal{V}^{|2|}\} \subseteq \mathcal{P}$ where path $\hat{p}_{st}$ connects source s to destination t, together with a corresponding system of *compatible weights* w so that

- each path $\hat{p}_{st} \in \hat{\mathscr{P}}$ is the unique w-shortest path from s to t in graph $\mathscr{G}$.
- when the whole demand volume d_{st} of demand (s,t) is allocated to path $\hat{p}_{st}$ then for each link the resulting link load does not exceed link capacity:

$$\Sigma_{(s,t)\in\hat{\mathscr{D}}(e)}d_{st} \le c_e, \qquad e \in \mathscr{E}, \tag{8.1}$$

where $\hat{\mathscr{D}}(e) = \{(s,t) \in \mathscr{V}^{|2|} : \hat{p}_{st} \ni e\}$.

This routing version is called unique, unsplittable, or single shortest path routing in the literature.

Observe that the uniqueness of the shortest paths in a routing pattern is not common, since most weight systems in general induce more than one shortest path between a pair of nodes—consider for example the hop-count weight system $w_e = 1$. If non-uniqueness is the case, i.e., if the system of administrative weights w induces more than one shortest path for demand $(s,t) \in \mathscr{V}^{|2|}$, then the volume d_{st} is split, according to the so-called *ECMP rule*, among all the shortest paths from source s to destination t. We note here that other ways of handling the shortest path non-uniqueness are sometimes considered, as, for example, selecting one particular path among all the shortest paths at random. When used in a real network, however, such rules lead to traffic routing that is different from the traffic pattern assumed at the weight design stage. Therefore, they are excluded from further considerations.

ECMP (Equal Cost Multi-path) is a specific rule to split traffic among shortest paths: at each node $v \in \mathscr{V}$ the total flow X_{vt} from v destined to any node $t \in \mathscr{V}$, $v \ne t$ (X_{vt} is composed of traffic transited via v and originated at v), is split equally among all the links outgoing from node v that belong to the w-shortest paths from v to t. This rule is illustrated in Figure 8.1 for a weight system with all link weights equal to 1 ($w_e = 1$) and one demand (s,t) with $d_{st} = 1$. There are three shortest paths from s to t: $s-a-c-t$, $s-a-d-t$, and $s-b-e-t$. According to ECMP, the flow in node s is split into two equal parts, the flow in node a also into two, and in the remaining nodes the flow is not split. As the result we obtain the following link flows: $x_{sa} = x_{sb} = x_{be} = x_{et} = \frac{1}{2}$, $x_{ac} = x_{ad} = x_{ct} = x_{dt} = \frac{1}{4}$. Observe that if we had reversed the directions of all links and of the demand then the resulting ECMP link flows would be equal to $x_{tc} = x_{td} = x_{te} = x_{ca} = x_{da} = x_{eb} = x_{bs} = \frac{1}{3}$, $x_{as} = \frac{2}{3}$.

Let $x_e(w)$ denote the ECMP flow induced by system w on link $e \in \mathscr{E}$. For a fixed network with given demand volumes, the flows $x_e(w)$, $e \in \mathscr{E}$, depend only on the
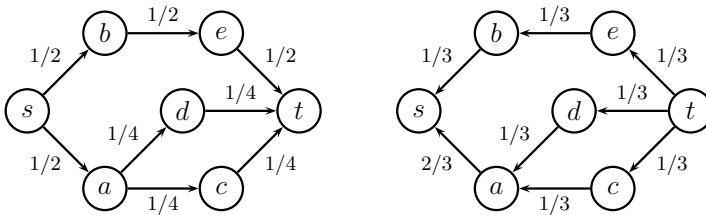

Fig. 8.1 Illustration of the ECMP rule

weight system w. If the weight system induces unique shortest paths, then each flow $x_e(w)$ is equal to the left-hand side of inequalitie (8.1). Imposing an upper bound on the link weights, the STEP problem for the ECMP rule can informally be stated as follows.

$$
\begin{align}
\text{find} \qquad & w = (w_e : e \in \mathscr{E}) \tag{8.2a} \\
\text{minimizing} \qquad & Z \tag{8.2b} \\
\text{subject to} \qquad & x_e(w) \le Z c_e, & e \in \mathscr{E}, \tag{8.2c} \\
& w_e \in \{1, 2, \dots, K\}, & e \in \mathscr{E}. \tag{8.2d}
\end{align}
$$

We note that objective (8.2b) minimizes the maximum link utilization—a criterion related to link congestion. Other commonly used traffic engineering objectives are discussed in Section 8.7.2. The above formulation is informal since the flows $x_e = x_e(w)$, $e \in \mathscr{E}$, assigning the ECMP flows to the links for a given weight system w, are not explicitly specified. In fact, the mapping of w to the induced flows $x_e(w)$, $e \in \mathscr{E}$, is not straightforward, as explained in Section 8.5.3. The computation of flows becomes simpler for the unsplittable shortest path version of STEP, as all traffic volume of a demand then is assigned to its unique shortest path.

An important subproblem of STEP is the so-called *inverse shortest path problem*, abbreviated ISP. It consists of finding a weight system w that induces precisely the assumed routing pattern $\hat{\mathscr{P}}$. Different variants of ISP are considered for routing patterns with unique shortest paths $\hat{\mathscr{P}} = \{\hat{p}_{st} : (s,t) \in \mathscr{V}^{|2|}\}$ and for the patterns admitting multiple shortest paths for each demand (applied together with the ECMP rule).

8.2.3 Discussion

The shortest path traffic engineering problem STEP described by the informal formulation (8.2) is known to be very difficult and, needless to say, is NP-hard in both ECMP and unsplittable versions [11, 45, 54], and is also NP-hard to approximate [11, 45].

Exact solutions of STEP can be achieved with (mixed) integer programming methods. Using formulations that use binary routing variables to indicate whether a link belongs to a shortest path or not (together with flow variables, path length variables, and weight variables), such as model (8.18) in Subsection 8.7.1, STEP can in principle be solved to optimality. Models of this kind have been considered by many authors (see Section 8.8); unfortunately, the relation between the shortest paths and the routing weights always leads to quadratic or very large big-M models, which are computationally extremely hard and not suitable for solving real-world problems.

Because of the inherent difficulty of STEP, various heuristic approaches for the solution of network design and routing problems in shortest path networks have been

proposed. Algorithms using local search, simulated annealing, or Lagrangian relaxation techniques with the routing weights as primary decision variables have been introduced by several authors (see Section 8.8). Such methods usually work quite well; still, they cannot guarantee optimal solutions nor do they provide means to estimate the quality of the solution (which is the case for the integer-programming-based approaches). In Section 8.5 we discuss the basic ideas and main challenges for successful implementations of such heuristics.

The inverse shortest paths problem (ISP) can be formulated as a linear program (see Section 8.3.2), and, as such, solved in polynomial time. Even if the weights are required to take nonnegative integer values, ISP remains polynomial, because any set of non-integer weights can be scaled easily to (possibly very large) integer weights. There is also a simple rounding scheme (see [6]) that produces integer weights which exceed the smallest possible maximum weight by the factor of at most $\min(|\mathcal{V}|/2, |P_{\max}|)$, where $P_{\max}$ is the longest shortest path in the considered routing pattern. The problem of finding the smallest maximum weight or weights not exceeding a given upper bound, however, is NP-hard [12].

Finally, we mention that a lot of work has been devoted to finding necessary conditions for a given set of routing paths to be compatible with a set of link weights. As these conditions play a crucial role in resolution methods for STEP, we will come back to them in Subsection 8.3.2 and Section 8.4 (see also Section 8.8 for historical remarks).

SPR is (heavily) constrained by the shortest path requirement and hence inherently less efficient in bandwidth utilization than other, less constrained routing strategies. Because of that, ECMP routing is in general less traffic efficient than its fractional multi-commodity flow routing counterpart, which permits us to split each demand's traffic arbitrarily among all paths between the demand's end nodes. Similarly, unsplittable shortest path routing is inferior to unsplittable multi-commodity flow routing, which must send each demand's traffic unsplit along a single path, but may choose the paths for different demands independently of each other. A question that arises naturally is whether the performance gap between an optimized shortest path routing and another, less constrained optimized routing is important or not. Bley [11] presented several classes of examples where the best link utilization that can be obtained with unsplittable shortest path routing exceeds the utilization obtained with unsplittable flow routing by a factor of $\Omega(|\mathcal{V}|^2)$. On the other hand, Fortz and Thorup [42] showed that for many real-world communication networks the gap between the ECMP shortest path routing version and the optimal fractional multi-commodity flow routing is virtually negligible. Similar observations have been reported by Ben Ameur et al. [7]. Still, if survivability issues and rerouting are taken into account, the gap can increase significantly.

8.3 Integer Programming Approach

In this section, we present a two-phase approach developed in [13, 18, 20, 46, 52, 73, 78]. The approach has been successfully used in the planning of the German national education and research network for several years [17, 19]. Similarly to Benders' decomposition, it decomposes the problem of finding an optimal shortest path routing into the master problem of finding the optimal end-to-end paths in the first phase, and the client problem of finding compatible routing weights for these paths in the second phase. An efficient version of the two-phase approach is achieved when the iterative use of the two phases is embedded in a branch-and-cut algorithm.

The master problem is formulated as an integer linear program and solved with a branch-and-cut algorithm [63, 83]. Instead of using routing weight variables, the underlying formulation contains special inequalities to exclude invalid routing patterns, i.e., path sets that do not correspond to the set of all shortest paths for any weight system. These inequalities are generated dynamically as cutting planes by the client problem during the execution of the branch-and-cut algorithm.

8.3.1 Optimizing the Routing Paths

There are several ways to formulate the master problem of STEP as a mixed-integer linear program. In the following, we present an aggregated node-link formulation of STEP for the ECMP routing version.

The primary decision variables in this formulation are variables $u_{et} \in \{0, 1\}$ for all $t \in \mathcal{V}$ and $e \in \mathcal{E} \setminus \delta^+(t)$. These variables describe the so-called shortest path graphs (SP graphs) to all destinations $t \in \mathcal{V}$. Each variable u_{et} is supposed to be equal to 1 if, and only if, there is a shortest path from node $a(e)$ to node t that contains link e.

In addition, the model uses variables $x_{et} \in \mathbb{R}$ for all $t \in \mathcal{V}$ and $e \in \mathcal{E} \setminus \delta^+(t)$ and $z_{vt} \in \mathbb{R}$ for all $t \in \mathcal{V}$ and $v \in \mathcal{V} \setminus \{t\}$, and a single variable $Z \in \mathbb{R}$. Variable x_{et} expresses the aggregated traffic flow that is sent across link e from all possible origins towards destination t. If at some node v the aggregated flow towards t is equally split and sent via several shortest paths, then this common flow value is represented by variable z_{vt}. In effect, the same amount z_{vt} of flow is sent across all links $e \in \delta^+(v)$ that belong to at least one of these shortest paths. Finally, variable Z represents the maximum link utilization (congestion).

With these variables the master problem is formulated as follows:

STEP Master
find

u_{et} binary variable indicating whether link e is on a shortest path to node t

x_{et} flow destined to node t on link e

z_{vt} flow destined to node t leaving node v on links on the shortest paths to t

Z maximum link utilization/congestion

minimize

$$Z \tag{8.3a}$$

subject to

$$\sum_{e \in \delta^+(v)} x_{et} - \sum_{e \in \delta^-(v)} x_{et} = d_{vt} \qquad t \in \mathcal{V}, v \in \mathcal{V} \setminus \{t\} \tag{8.3b}$$

$$\sum_{t \in \mathcal{V} \setminus \{b(e)\}} x_{et} \le c_e Z \qquad e \in \mathcal{E} \tag{8.3c}$$

$$x_{et} \le (D_t - d_{b(e)t}) u_{et} \qquad t \in \mathcal{V}, e \in \mathcal{E} \setminus \delta^+(t) \tag{8.3d}$$

$$0 \le x_{et} - z_{a(e)t} \le (D_t - d_{b(e)t})(1 - u_{et}) \qquad t \in \mathcal{V}, e \in \mathcal{E} \setminus \delta^+(t) \tag{8.3e}$$

$$\sum_{(e,t) \in C} u_{et} - \sum_{(e,t) \in \bar{C}} u_{et} \le |C| - 1 \qquad (C, \bar{C}) \in \mathcal{C} \tag{8.3f}$$

$$u_{et} \in \{0, 1\} \qquad t \in \mathcal{V}, e \in \mathcal{E} \setminus \delta^+(t) \tag{8.3g}$$

$$x_{et} \ge 0 \qquad t \in \mathcal{V}, e \in \mathcal{E} \setminus \delta^+(t) \tag{8.3h}$$

$$z_{vt} \ge 0 \qquad t \in \mathcal{V}, v \in \mathcal{V} \setminus \{t\} \tag{8.3i}$$

$$Z \ge 0. \tag{8.3j}$$

Subproblem (8.3a)–(8.3c) is an aggregated node-link formulation of a capacitated multi-commodity flow problem with aggregated flows x, whose objective is to minimize the maximum link utilization Z. The next two constraints express the ECMP traffic splitting rule. Inequality (8.3d) forces traffic destined to node t to use only the links that are chosen to be shortest path links, i.e., links $e \in \mathcal{E}$ with $u_{et} = 1$. Constraint (8.3e) ensures that in each node the traffic to destination node t is split equally among the links assigned to that destination.

Finally, the "conflict" constraints (8.3f) ensure that each integer solution of (8.3) is an admissible ECMP routing. Let $C, \bar{C} \subset \mathcal{V} \times \mathcal{E}$ be two disjoint sets of node-link pairs. If there exists no ECMP routing such that e is on a shortest path towards t for all $(e,t) \in C$, and e is not on a shortest path towards t for all $(e,t) \in \bar{C}$, we say that the pair $(C, \bar{C})$ is an ECMP conflict. The family of all such conflicts is denoted by $\mathcal{C}$. Constraints (8.3f) ensure that no solution simultaneously contains all the shortest-path links and all the non-shortest-path links of any such conflict. This implies that any integer solution of (8.3) is indeed an admissible shortest path routing pattern. Figure 8.2 on page 212 illustrates three special types of these conflict constraints. We discuss the conflict constraints in more detail in Section 8.4.

In general, the number of conflict constraints (8.3f) can be exponentially large and the structure of the conflicts can be extremely complicated [13]. Therefore, only

few of these constraints are included in the model initially. The majority of them are separated via the client problem in the branch-and-cut solution process.

8.3.2 Finding Compatible Routing Weights

Now suppose we have a solution (u,x,z,Z) of formulation (8.3) or, more precisely, of a subsystem of (8.3), which contains only those conflict constraints (8.3f) that have been generated so far in the process, and not all conflict constraints. For each destination $t \in \mathcal{V}$, the values of the binary variables u in this solution define a link set $A_t = \{e \in \mathcal{E} : u_{et} = 1\}$. The link set A_t is called the *shortest path graph (SP graph) for destination t*. For each $t \in \mathcal{V}$, we want the links $e \in A_t$ to be on a shortest path from $a(e)$ to t and the links $e \notin A_t$ to be not on a shortest path from $a(e)$ to t. Link weights $w_e \in \mathbb{R}_+$, $e \in \mathcal{E}$, for which these conditions hold are said to be *compatible* with the given SP graphs A_t.

Our goal in the client problem is to find compatible link weights w_e, $e \in \mathcal{E}$, for the SP graphs given by the master problem's solution. However, if the given solution (u,x,z,Z) violates some of the conflict constraints that have not yet been added to the master formulation (8.3), then such link weights do not exist. In this case, the task is to generate (at least) one of these violated inequalities.

The first part of this problem is just the inverse shortest paths problem, and can be solved with linear programming techniques. A number of alternative formulations for ISP have been proposed in the literature [6, 13, 68]. In the following, we present an aggregated formulation for ISP, which fits very nicely into the aggregated formulation of the master problem. It uses a variable $w_e \in \mathbb{Z}$ for the weight of each link $e \in \mathcal{E}$, a variable $w_{\max} \in \mathbb{Z}$ for the maximum of these weights, and a variable $r_{vt} \in \mathbb{R}$ for the potential of each node $v \in \mathcal{V}$ with respect to each destination $t \in \mathcal{V}$ and the weights w. (If $r_{tt} = 0$, the smallest possible potential r_{vt} of node v is exactly the distance from v to t with respect to the link weights w.) With these variables, the inverse shortest paths problem for the given SP graphs A_t, $t \in \mathcal{V}$, can be formulated as follows:

ISP Client
find

$\qquad w_e \qquad$ routing weight of link e

$\qquad w_{\max} \qquad$ maximum routing weight

$\qquad r_{vt} \qquad$ potential of node v with respect to destination t and weights w

minimize

$$w_{\max} \tag{8.4a}$$

subject to

$$w_e - r_{a(e)t} + r_{b(e)t} = 0 \qquad\qquad t \in \mathcal{V}, e \in A_t \qquad\qquad (8.4b)$$

$$w_e - r_{a(e)t} + r_{b(e)t} \geq 1 \qquad\qquad t \in \mathcal{V}, e \notin A_t \qquad\qquad (8.4c)$$

$$1 \leq w_e \leq w_{\max} \qquad\qquad e \in \mathcal{E} \qquad\qquad (8.4d)$$

$$r_{vt} \in \mathbb{R} \qquad\qquad t \in \mathcal{V}, v \in \mathcal{V} \qquad\qquad (8.4e)$$

$$w_e \in \mathbb{Z} \qquad\qquad e \in \mathcal{E}. \qquad\qquad (8.4f)$$

Constraints (8.4b) and (8.4c) (together with nonnegativity of the weight variables w_e implied by (8.4d)) ensure that the weights w_e in any solution of formulation (8.4) are compatible with the given SP graphs. The quantity $w_e - r_{a(e)t} + r_{b(e)t}$ measures the difference between the length of the shortest path which starts in node $a(e)$, goes over link e, and ends in node t, and the distance from the node of $a(e)$ to t. This difference must be 0 for all links that are supposed to be on a shortest path and strictly greater than 0 for all links that are supposed to be not on a shortest path, as expressed in constraints (8.4b) and (8.4c). Hence, formulation (8.4) has a solution if and only if there exist compatible weights for the given family of SP graphs A_t, $t \in \mathcal{V}$. Furthermore, there are compatible weights in the range $\{1, 2, \ldots, K\}$ if and only if the optimal solution value $w_{\max}$ of formulation (8.4) is less than or equal to K.

Note that formulation (8.4) is an integer program and may be computationally hard. In fact, Bley [12] proved that it is already NP-hard to approximate its optimum within a factor less than $9/8$ in general.

In our decomposition approach, it is sufficient to solve only the linear relaxation of (8.4) and scale and round its optimal fractional solution to an integer-feasible solution of (8.4). It is not difficult to verify that the integer program (8.4) has a solution if and only if its linear relaxation does. Using the rounding scheme proposed by Ben-Ameur and Gourdin [6], we obtain weights that exceed the minimal ones by a factor of at most $\min(|\mathcal{V}|/2, |P_{\max}|)$, where $P_{\max}$ is the longest prescribed shortest path. For practically relevant network sizes, the weights computed with this approximate method easily fit into the admissible range of all modern routing protocols. So, we can safely ignore the integrality constraint (8.4f) in practice.

If the linear relaxation of (8.4) is infeasible, then the given solution (u, x, z, Z) of the (incomplete) master formulation is not a valid ECMP routing. In this case, the presumed routing contains at least one conflict $(C, \bar{C}) \in \mathcal{C}$ with $C \subseteq \{(e, t) : u_{et} = 1\}$ and $\bar{C} \subseteq \{(e, t) : u_{et} = 0\}$, whose corresponding conflict inequality (8.3f) is violated by the given solution (u, x, z, Z). Adding this inequality to the master formulation, one can cut off the current invalid solution.

In practice it is important to generate conflicts with small sets C and $\bar{C}$, as this leads to stronger inequalities (8.3f). Inclusion-wise minimal conflicts, i.e., conflicts $(C, \bar{C}) \in \mathcal{C}$ such that there is no other conflict $(C', \bar{C}') \in \mathcal{C}$ with $C' \subseteq C$ and $\bar{C}' \subseteq \bar{C}$, can be computed in polynomial time using simple greedy techniques in combination with a generalized version of the above linear programming formulation. Finding a conflict of minimum total size $|C| + |\bar{C}|$, however, is NP-hard [13].

In Sections 8.4.1 and 8.4.2, we describe several subclasses of the conflict inequalities (8.3f) that are separable in polynomial time. An algorithm to generate strongly

violated conflict inequalities (8.3f) based on an (approximate) integer programming formulation of the separation problem is presented in Section 8.4.3.

For the efficiency of the overall solution approach, it is important to consider the client problem not only for those solutions of the master problem where all variables u_{et} are integer, but also for those solutions where some of these variables are fractional. For each $t \in \mathcal{V}$, the values of the variables u in the current solution of the master problem define a second link set $\bar{A}_t := \{e \in \mathcal{E} : u_{et} = 0\}$. The pair of the two link sets $(A_t, \bar{A}_t)$ is called the *partial SP graph for destination t*. If all variables u_{et} are integer, then we obviously have $A_t \cup \bar{A}_t = \mathcal{E} \setminus \delta^+(t)$ for all $t \in \mathcal{V}$. For each $t \in \mathcal{V}$, we want the links $e \in A_t$ to be on a shortest path from $a(e)$ to t and the links $e \in \bar{A}_t$ to be not on a shortest path from $a(e)$ to t. Links $e \in \mathcal{E}$ that are neither in A_t nor in $\bar{A}_t$ may or may not be on a shortest path from $a(e)$ to t. Routing weights $w_e \in \mathbb{R}_+$, $e \in \mathcal{E}$, that satisfy these conditions are said to be *compatible* with the given partial SP graphs $(A_t, \bar{A}_t)$, $t \in \mathcal{V}$.

The techniques presented above for the inverse shortest paths problem with complete SP graphs generalize straightforwardly to the inverse shortest paths problem with partial SP graphs. Replacing inequality (8.4c) in formulation (8.4) with the inequalities

$$w_e - r_{a(e)t} + r_{b(e)t} \geq 1 \qquad\qquad t \in \mathcal{V}, e \in \bar{A}_t \qquad\qquad (8.4\text{c'})$$

and

$$w_e - r_{a(e)t} + r_{b(e)t} \geq 0 \qquad\qquad t \in \mathcal{V}, e \in \mathcal{E} \setminus (A_t \cup \bar{A}_t), \qquad (8.4\text{c''})$$

we obtain an integer linear model for the inverse shortest paths problem with partial SP graphs. Solving its linear relaxation, we can again decide in polynomial time whether the given partial SP graphs can be extended to a valid shortest path routing or not. If the linear relaxation has a solution, the same scaling and rounding approach as that for complete SP graphs can be used to compute reasonably small integer weights that are compatible with the given partial SP graphs. Otherwise, the solution of the current incomplete master formulation can be cut off by a conflict inequality (8.3f) derived from the dual relaxation of the inverse shortest paths problem, even though not all SP variables u_{et} are integer yet. Each assignment of 0/1 values to those variables u_{et} that are not integer yet would lead to an invalid routing.

8.4 Shortest Path Routing Inequalities

As we have already seen in the previous section, discovering and generating proper cuts (valid inequalities) is crucial for effectiveness of integer programming approaches to STEP. In this section we will study this issue in more detail. We first derive a set of cuts that follow from combinatorial/structural properties of shortest paths. Then we discuss unobtainable cycles—a strong necessary condition on a

set of routing paths to posses a compatible weight system. Finally, we show how the formulations related to unobtainable cycles can be used to derive general valid inequalities, eliminating inconsistent routing patterns from the master problem discussed in Subsection 8.3.1.

8.4.1 Combinatorial Cuts

In this section we will discuss deriving inequalities of the form (8.3f) directly from the combinatorial properties of shortest paths. We discuss three such basic properties [71], called transit, split, and cycle (see Figure 8.2). These properties are derived from the subpath consistency (also called Bellman property) of shortest paths and describe the consistency conditions of shortest paths between different pairs of nodes. The resulting inequalities either extend or generalize the types of combinatorial cuts that were proposed in [20, 48] in the context of ECMP routing .

$$\sum_{f \in \mathscr{P}_{b(e)t}} (1 - u_{fv}) + (1 - u_{ev}) + u_{et} \geq 1$$

$$\sum_{f \in \mathscr{P}_{st}} (1 - u_{fv}) + \sum_{g \in \mathscr{Q}_{ts}} (1 - u_{gv}) \geq 1$$

$$\sum_{f \in \mathscr{P}_{b(e)t}} (1 - u_{fv}) + (1 - u_{ev}) + (1 - u_{gt}) + u_{gv} \geq 1$$
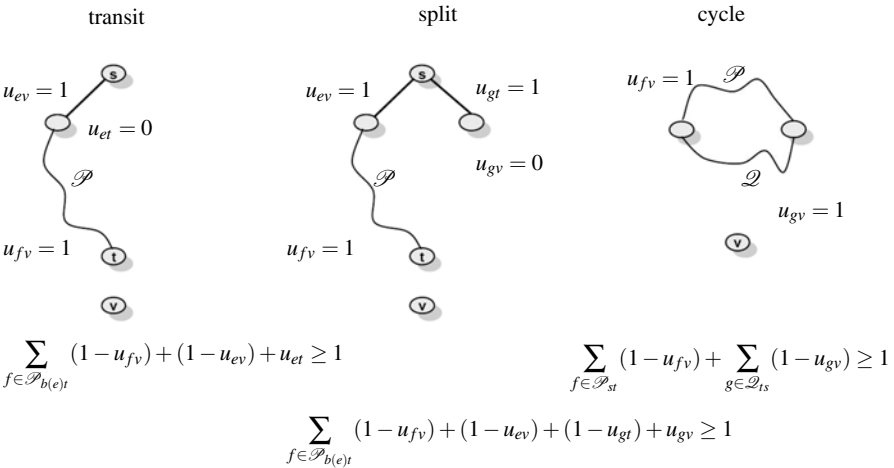
Fig. 8.2 Different types of combinatorial valid inequalities

The *transit* property expresses a relation between shortest paths to a destination and shortest paths to the transit nodes of those paths to the destination. Assume that there is a path from node s to node t (such a path can be decomposed into the starting link e and path $\mathscr{P}_{b(e)t}$ from $b(e)$ to t), with all links belonging to shortest paths to some node v. Thus, t is a transit node on a shortest path from s to v. Then, all links between s and t on the path to v, in particular link e, must belong to a shortest path to t, since a shorter path from s to t should have been used to v as well. The following inequality separates vectors u which contradict this property:

$$\sum_{f \in \mathscr{P}_{b(e)t}} (1 - u_{fv}) + (1 - u_{ev}) + u_{et} \geq 1. \tag{8.5}$$

To find the most violated inequality (8.5) for given s, v, t, and e, it is enough to find a shortest path $\mathscr{P}_{b(e)t}$ from $b(e)$ to t using the values $1 - u_{ev}$ for the link weights. This inequality is stronger than the one used in [48], because it skips the values of the variables defining shortest paths from t to v, and considers the path from s to t and not from s to v.

The *split* property expresses a relation between splitting traffic among shortest paths to a destination and splitting traffic among shortest paths to the transit nodes. Assume the same situation as in the first case, and, additionally, that there is another link g originating in node s which is on a shortest path to node t. Then g must be on a shortest path to v, because there are two paths of equal length from s to t (one starting with link g and one with link e because of the *transit* property), and as one is used to reach v, the other should be used to reach v as well. The vectors u that contradict this property can be separated using the following inequality:

$$\sum_{f \in \mathscr{P}_{b(e)t}} (1 - u_{fv}) + (1 - u_{ev}) + (1 - u_{gt}) + u_{gv} \geq 1. \tag{8.6}$$

To find the most violated inequality (8.6) for given s, v, t, e, and g it is enough to find the same path $\mathscr{P}_{b(e)t}$ as in the case of the *transit* property.

The *cycle* property expresses a relation between shortest paths to a single destination. Since the values of link weights are strictly positive, on a shortest path to a destination the distances of the consecutive nodes to that destination are decreasing. Thus, the segments $\mathscr{P}_{st}$ and $\mathscr{Q}_{ts}$ of two such paths cannot form a cycle. The following inequality separates vectors u which contradict this property:

$$\sum_{f \in \mathscr{P}_{st}} (1 - u_{fv}) + \sum_{g \in \mathscr{Q}_{ts}} (1 - u_{gv}) \geq 1. \tag{8.7}$$

To find the most violated inequality (8.7) for given s, t, and v, it is enough to find a pair of shortest paths, from s to t, and from t to s, respectively, using the values $1 - u_{ev}$ as the link weights.

To find all violated inequalities of types (8.5)–(8.7) for a given vector u, it is thus sufficient to determine, for each destination node v, the shortest paths between all pairs of nodes, using values $1 - u_{ev}$ as link weights; thus, the entire process has overall complexity of $\mathscr{O}(|\mathscr{V}|^4)$.

A number of combinatorial inequalities for unsplittable shortest path routing are discussed in [6, 13, 18, 49, 73, 77]. Similarly to (8.5), the following three inequalities express the transit property for unique shortest paths:

$$x_{av}^s - x_{at}^s + \sum_{e \in \delta^-(v)} x_{et}^s \leq 1 \qquad s, v, t \in \mathcal{V}, a \in \mathcal{E} \qquad (8.8a)$$

$$x_{at}^v - x_{at}^s + \sum_{e \in \delta^-(v)} x_{et}^s \leq 1 \qquad s, v, t \in \mathcal{V}, a \in \mathcal{E} \qquad (8.8b)$$

$$\frac{1}{2}(x_{av}^s + x_{at}^v - x_{at}^s) + \sum_{e \in \delta^-(v)} x_{et}^s \leq 1 \qquad s, v, t \in \mathcal{V}, a \in \mathcal{E} \qquad (8.8c)$$

These inequalities are formulated using non-aggregated flow variables x_{et}^s. Variable x_{et}^s defines the fraction of flow originated in s and destined to t on link e. Then, if $\sum_{e \in \delta^-(v)} x_{et}^s$ is greater than 0 (in case of unsplittable routing it is simply 1), node v must be a transit node on a shortest path from s to t. But then, if, for instance, the flow originated in s and destined to v uses some edge a, that edge must also be used by the flow originated in s and destined to t, which is exactly the meaning of inequality (8.8a).

To evaluate the effectiveness of the combinatorial inequalities we have performed numerical experiments (cf. Section 8.4.3) based on solving the integrated MIP problem of routing and link weight optimization with a commercial MIP solver, applying its regular B&C procedure in two settings: using and not using user-defined cuts. When no user-defined cuts were used, after $4,155$ seconds, visiting $664,000$ B&C nodes, and generating 382 standard cuts the computation was aborted as it ran out of memory. In contrast, when combinatorial inequalities were used as user-defined cuts, having generated the total of $1,556$ user-defined cuts and only 100 standard cuts, after 548 seconds the computation reached the optimum, reducing the number of visited B&C nodes to $68,700$.

8.4.2 Valid Cycles

In order to find an interesting class of SP graph conflicts, we focus on whether or not (8.4b)–(8.4e) has a feasible solution. Ignoring the boundedness and integrality of the weights, and letting γ_e^t be the dual variables to constraint sets (8.4b) and (8.4c), we get the following LP-dual after some reformulations (including eliminating the dual variables of $w_e \geq 1$).

$$\min \qquad \sum_{t \in \mathcal{V}} \sum_{e \in A_t} \gamma_e^t \qquad\qquad\qquad (8.9a)$$

$$\text{s.t.} \qquad \sum_{t \in \mathcal{V}} \gamma_e^t \leq 0 \qquad\qquad e \in \mathcal{E} \qquad (8.9b)$$

$$\sum_{e \in \delta^+(v)} \gamma_e^t - \sum_{e \in \delta^-(v)} \gamma_e^t = 0 \qquad t \in \mathcal{V}, v \in \mathcal{V} \qquad (8.9c)$$

$$\gamma_e^t \geq 0 \qquad\qquad e \in \mathcal{E} \setminus A_t, t \in \mathcal{V} \qquad (8.9d)$$

Model (8.9a)–(8.9d) can be seen as a *multi-commodity network flow problem*, with commodities indexed by t. Some flow may be negative, since γ_e^t may be negative for $e \in A_t$. Starting at the feasible solution $\gamma = 0$, we look for an unbounded solution to (8.9a)–(8.9d) , which would indicate that (8.4b)–(8.4e) is infeasible. Due to constraints (8.9c), each commodity must be changed in cycles, if at all. Due to constraints (8.9b), any increase of one commodity must be compensated for by a decrease of another commodity.

An arc can be used forwards (positive flow) or backwards (negative flow). Consider a cycle $C \subseteq \mathscr{E}$, $C = F \cup B$, where F means forwards and B backwards (for commodity l'). The following is a possible change.

$$\gamma_{ij}^{l'} = \theta \; (i,j) \in F, \; \gamma_{ij}^{l'} = -\theta \; (i,j) \in B, \; \gamma_{ij}^{l''} = -\theta \; (i,j) \in F, \; \gamma_{ij}^{l''} = \theta \; (i,j) \in B \; (8.10)$$

Let us now define some notation. A cycle $C = F \cup B$ is called *feasible* if $B \subseteq A_{t'}$ and $F \subseteq A_{t''}$, i.e., the arcs in B lie in one SP graph and the arcs in F lie in another. The arc (i,j) is called *eligible* if $(i,j) \in (F \setminus A_{t'}) \cup (B \setminus A_{t''})$. In words, an eligible arc lies in F but not in $A_{t'}$ or in B but not in $A_{t''}$.

A cycle $C = F \cup B$ is called *valid* if there exist two indices l' and l'' such that the cycle is feasible and contains at least one eligible arc.

We find that flow of commodities l' and l'' can be changed infinitely in a valid cycle, and the objective function value (8.9a) tends towards infinity. In other words, a valid cycle represents an unbounded solution to (8.9a)–(8.9d) .

Proposition 8.1. *If there exists a valid cycle, then there exists no compatible set of weights.*

See [28] for a proof for the spanning case. In [29] the authors prove that proposition 8.1 holds even if the SP graphs are not spanning, although then the model (8.9a)–(8.9d) cannot be used.

Letting $S_l(s,t)$ denote all subpaths from node s to node t in A_l; we say that A_k and A_l are *subpath consistent* if $S_k(s,t) = S_l(s,t)$ for all $s \in \mathscr{V}$ and $t \in \mathscr{V}$ such that $S_k(s,t) \neq \emptyset$ and $S_l(s,t) \neq \emptyset$. It is well-known that subpath consistency between all SP graphs is a necessary condition for the existence of compatible weights.

We can show the following.

- A valid cycle must contain at least three nodes and three arcs.
- If all SP graphs are trees, then any feasible cycle is also valid.
- If two SP graphs are subpath inconsistent, there exists a valid cycle.

Below, we give an example of subpath consistent SP graphs that have a valid cycle, and can conclude that the absence of a valid cycle is a stronger necessary condition for the existence of compatible weights than subpath consistency. In Figure 8.3 we give two SP graphs and the resulting valid cycle.

Considering the feasible set (8.9b)–(8.9d), we find the following (proved in [27]). A valid cycle represents an *extreme ray*. Valid cycles represent *all* extreme rays that only include *two* commodities. Valid cycles represent *all* extreme rays that use only *one cycle* and its reverse.
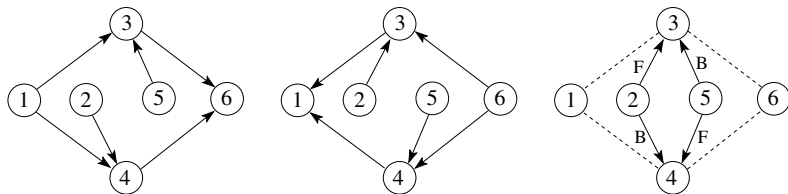
Fig. 8.3 Two SP graphs and a valid cycle

There are three possibilities for a set of SP graphs: compatible weights exist, valid cycles exist, or neither exists. In the last case, the unbounded solutions of (8.9a)–(8.9d) are of a more complicated structure than those represented by valid cycles, and require the usage of three or more commodities.

Computational tests suggest that the last case is not so common. In [28], a total of 1,423 different instances are solved. Of these 276 have compatible weights, 1,137 have valid cycles, and only 10 (i.e., 0.7%) have neither.

A method for finding valid cycles (called VC method) enumerates pairs of SP graphs, and constructs a graph $\bar{G}$, which contains the arcs in one SP graph and the reversed arcs in the other. One could then use *reductions* by removing nodes and arcs of $\bar{G}$ that cannot be a part of a valid cycle, and then try to find a feasible cycle containing a certain eligible arc. If there exists no such cycle, the eligible arc is removed. This is repeated until the whole graph is eliminated or a valid cycle is found. If the graph is completely eliminated by the reductions, there exists no valid cycle with the two SP graphs considered.

Since a valid cycle exists if and only if a strongly connected component of $\bar{G}$ contains an eligible arc, one can also search for strongly connected components, and remove those with less than three nodes, and those without eligible arcs.

Proposition 8.2. *After a finite and polynomial number of steps, the VC algorithm will terminate, either with a valid cycle or a proof that no valid cycle exists.*

Proposition 8.2 holds even if the SP graphs are not spanning. See [28] and [29] for details. The best complexity of a VC method is $\mathcal{O}(m^2|\mathcal{V}|^2)$, which reduces to $\mathcal{O}(m^2|\mathcal{V}|)$ if all SP graphs are trees.

Undirected paths can be converted into two SP graphs, one in each direction, and combined, so that the undirected single path case can be handled in a polynomial way by the VC method. This way, we have found a valid cycle in an instance that satisfies the generalized cyclic compatibility condition [6], so the cyclic conditions in [6] are not stronger necessary conditions for the existence of compatible weights than the absence of valid cycles.

If it is unknown if compatible weights exist, one can first try to find compatible weights by solving (8.4a)–(8.4e) with an LP code, and if one fails to find a feasible solution, one can proceed with the VC method. Another possibility is to first run the VC method, and if it fails to find a valid cycle, one can try to find the weights by solving (8.4a)–(8.4e). Computational tests reveal that for large problems, solving (8.4a)–(8.4e) as an LP might take more than 100 times longer than running the VC

method, so we recommend starting with the VC method, and only solving the LP problem (8.4a)–(8.4e) if no valid cycle exists.

Considering (8.4a), (8.4b), (8.4c'), (8.4c"), (8.4d), (8.4e), we find that the arcs in $\mathscr{E} \setminus (A_t \cup \bar{A}_t)$ will be included in (8.9a) and will also occur in (8.9d). This means that the arcs in $\mathscr{E} \setminus (A_t \cup \bar{A}_t)$ cannot be part of F or B, and are not eligible, and so play a very passive role in this context. Therefore, the VC methods are not changed.

Finally we note that in order to prohibit a certain valid cycle, one can introduce constraints that either make the cycle infeasible or remove the eligible arcs in the cycle.

8.4.3 General Inequalities

In the previous sections, we discussed special well-structured conflicts among SP graphs. In general, however, such conflicts may be extremely complex. Now, we will discuss methods that might discover such conflicts in a general case, and may also generate corresponding valid inequalities based on fractional solutions to problem (8.3). The following discussion is based on [78].

The set of all admissible binary vectors u will be denoted by $\mathscr{U}$. For each $e \in \mathscr{E}$ and $t \in \mathscr{V}$ consider the quantity $\delta_{et} = r_{b(e)t} + w_e - r_{a(e)t}$. Clearly, link e is on a shortest path to node t if, and only if, $\delta_{et} = 0$. Hence, a routing vector u defines a shortest path routing configuration if there exists a system w of positive weights such that $\delta_{et} \geq 1$ if $u_{et} = 0$, and $\delta_{et} = 0$ if $u_{et} = 1$. Since these conditions can be rewritten as $\delta_{et} + u_{et} \geq 1$, $\delta_{et} u_{et} = 0$, the following linear program in variables y, $w = (w_e : e \in \mathscr{E})$, and $r = (r_{st} : s, t \in \mathscr{V})$ can be used to check whether a given vector u defines admissible routing:

$$\mathbf{P}(u): \quad \min \qquad\qquad\qquad y \qquad\qquad\qquad\qquad\qquad (8.11a)$$

$$\text{s.t.} \qquad r_{b(e)t} + w_e - r_{a(e)t} + u_{et} \geq 1 - y \qquad e \in \mathscr{E}, t \in \mathscr{V}, \qquad (8.11b)$$

$$(r_{b(e)t} + w_e - r_{a(e)t})u_{et} \leq y \qquad e \in \mathscr{E}, t \in \mathscr{V}, \qquad (8.11c)$$

$$w_e \geq 1 \qquad e \in \mathscr{E}, \qquad (8.11d)$$

$$r_{tt} = 0 \qquad t \in \mathscr{V}, \qquad (8.11e)$$

$$y \geq 0. \qquad\qquad\qquad\qquad (8.11f)$$

This program is similar to (8.4); however in this case, being continuous, routing variables u_{et} are used explicitly in the constraints. Let $(w^*(u), r^*(u), y^*(u))$ denote an optimal solution of $\mathbf{P}(u)$. If $y^*(u) = 0$ then $w^*(u)$ and $r^*(u)$ satisfy introduced shortest path constraints, and hence $u \in \mathscr{U}$, i.e., u describes an admissible shortest path routing configuration. On the other hand, if $y^*(u) > 0$, there is no assignment of link weights which can generate the routing configuration u.

Now, consider the problem dual to $\mathbf{P}(u)$; let $\mu = (\mu_{et} : e \in \mathscr{E}, t \in \mathscr{V})$, $\pi = (\pi_{et} : e \in \mathscr{E}, t \in \mathscr{V})$, and $\theta = (\theta_e : e \in \mathscr{E})$ be the vectors of the dual variables corresponding

to constraints (8.11b), (8.11c), and (8.11d), respectively. Using some algebra, we eliminate dual variables θ_e, substitute variables μ_{ev} with new variables $\varphi_{et} = u_{et}\pi_{et} - \mu_{et}$, and finally obtain the following form of the dual problem expressed in variables $\varphi = (\varphi_{et} : e \in \mathscr{E}, t \in \mathscr{V})$ and $\pi = (\pi_{et} : e \in \mathscr{E}, t \in \mathscr{V})$:

$$\mathbf{F}(u): \quad \max \qquad F_u(\varphi) = \sum_{e \in \mathscr{E}} \sum_{t \in \mathscr{V}} u_{et}\varphi_{et} \tag{8.12a}$$

$$\text{s.t.} \qquad \sum_{t \in \mathscr{V}} \varphi_{et} \geq 0 \qquad e \in \mathscr{E} \tag{8.12b}$$

$$\sum_{e \in \delta+(v)} \varphi_{et} - \sum_{e \in \delta-(v)} \varphi_{et} = 0 \qquad v, t \in \mathscr{V} \tag{8.12c}$$

$$\sum_{e \in \mathscr{E}} \sum_{t \in \mathscr{V}} \varphi_{et} - \sum_{e \in \mathscr{E}} \sum_{t \in \mathscr{V}} (u_{et} + 1)\pi_{et} \geq -1 \tag{8.12d}$$

$$\varphi_{et} \leq u_{et}\pi_{et}, \ \pi_{et} \geq 0 \qquad e \in \mathscr{E}, t \in \mathscr{V}. \tag{8.12e}$$

Problem $\mathbf{F}(u)$ can be regarded as a special type of the multi-commodity flow problem with φ_{et} interpreted as an amount (bounded, and possibly negative) of (pseudo-) flow of commodity t on link e. Due to constraint (8.12c) the flow of each commodity is circular, and due to (8.12b) the total amount of flow on each link is nonnegative. The objective is to find the network flow with maximum total revenue where u_{et} can be interpreted as the unit revenue of using link e by commodity t. Problem $\mathbf{F}(u)$ is a generalization of problem (8.9) from the previous section, variables φ_{et} corresponding to variables γ_{et} (with reversed sign). The major difference results from the fact that with fractional routing vectors u, variables u_{et} explicitly appear in formulation (8.12).

Let F_u^* denote the optimal objective of (8.12). It holds that the problem is feasible, that $0 \leq F_u^* \leq 1$ for any u, and that a vector u defines an admissible shortest path routing configuration if, and only if, $F_u^* = 0$.

From the dual test, general valid inequalities that separate non-admissible routing vectors u can now be derived. Suppose that $\mathscr{I}^1$ and $\mathscr{I}^0$ are two disjoint sets of pairs $(e,t) \in \mathscr{E} \times \mathscr{V}$, i.e., $\mathscr{I}^1, \mathscr{I}^0 \subseteq \mathscr{E} \times \mathscr{V}$, and $\mathscr{I}^1 \cap \mathscr{I}^0 = \emptyset$. Let S denote a real-valued function defined as follows:

$$S(\mathscr{I}^1, \mathscr{I}^0; u) = \sum_{(e,t) \in \mathscr{I}^1} (1 - u_{et}) + \sum_{(e,t) \in \mathscr{I}^0} u_{et}. \tag{8.13}$$

We will consider valid inequalities of the following form:

$$S(\mathscr{I}^1, \mathscr{I}^0; u) \geq 1. \tag{8.14}$$

Note, that this is exactly the form of conflict-eliminating constraints (8.3f) with $\mathscr{I}^1 = C$ and $\mathscr{I}^0 = \bar{C}$.

For binary u^0 all terms in (8.13) are binary, so $S(\mathscr{I}^1, \mathscr{I}^0; u^0)$ is nonnegative and integer. If (8.14) is not satisfied, then $S(\mathscr{I}^1, \mathscr{I}^0; u^0) = 0$, and all its terms must be equal to 0. Hence, inequality (8.14) does not hold for a binary vector u^0 (i.e.,

$S(\mathscr{I}^1, \mathscr{I}^0; u) = 0$) if, and only if, $u_{et}^0 = 1$ for all $(e,t) \in \mathscr{I}^1$ and $u_{et}^0 = 0$ for all $(e,t) \in \mathscr{I}^0$. Suppose u^0 is a binary vector defining a non-admissible routing configuration, and (φ, π) is a solution of problem $\mathbf{F}(u^0)$ such that $F_{u^0}(\varphi) > 0$. Then, inequality (8.14) with $\mathscr{I}^1 = \mathscr{I}^1_+(u^0) = \{(e,t) \in \mathscr{E} \times \mathscr{V} : u_{et}^0 = 1 \wedge \varphi_{et} > 0\}$ and $\mathscr{I}^0 = \mathscr{I}^0_-(u^0) = \{(e,t) \in \mathscr{E} \times \mathscr{V} : u_{et}^0 = 0 \wedge \varphi_{et} < 0\}$ separates u^0 and does not separate any admissible routing vector u (i.e., any binary $u \in \mathscr{U}$).

Fractional vector u^0 (u with at least one strictly fractional u_{et} always describes a non-admissible shortest path routing configuration) can also be separated with a general method analogous to the one used for separating non-admissible binary shortest path routing vectors u by solving problem $\mathbf{F}(u^0)$. Separating fractional vectors is however more complex. The goal is to find sets $\mathscr{I}^1$ and $\mathscr{I}^0$ for which inequality (8.14) separates u^0 and does not separate any admissible routing configuration, and for which these sets determine the most violated valid inequality of the considered type. Let $q = (q_{et} : e \in \mathscr{E}, v \in \mathscr{V})$ be a binary vector and define $\mathscr{J}(q) = \{(e,t) \in \mathscr{E} \times \mathscr{V} : q_{et} = 1\}$ (i.e., q is the characteristic function of set $\mathscr{J}(q)$). Suppose that $\mathscr{I}^1 = \mathscr{J}(y)$ and $\mathscr{I}^0 = \mathscr{J}(z)$ for two binary vectors $y = (y_{et} : e \in \mathscr{E}, t \in \mathscr{V})$ and $z = (z_{et} : e \in \mathscr{E}, t \in \mathscr{V})$. We assume that sets $\mathscr{I}^0$ and $\mathscr{I}^1$ are disjoint, so $y_{et} + z_{et} \leq 1$ must hold for all pairs (e,t). Then, function (8.13) can be calculated as

$$S(\mathscr{I}^1, \mathscr{I}^0; u) = \sum_{e \in \mathscr{E}} \sum_{t \in \mathscr{V}} ((1 - u_{et}) y_{et} + u_{et} z_{et}). \tag{8.15}$$

Now, for a non-admissible u^0, an issue arises of how to determine such vectors y and z, such that $S(\mathscr{J}(y), \mathscr{J}(z); u^0) < 1$ and all binary vectors u separated by (8.14) correspond to non-admissible routing configurations. Consider the following problem (where Δ is a small strictly positive constant):

$$\mathbf{G}(u): \quad \min \quad G_u(y,z) = \sum_{e \in \mathscr{E}} \sum_{t \in \mathscr{V}} ((1 - u_{et}) y_{et} + u_{et} z_{et}) \tag{8.16a}$$

$$\text{s.t.} \quad (\varphi, \pi) \in \mathscr{F}_u \tag{8.16b}$$

$$\sum_{e \in \mathscr{E}} \sum_{t \in \mathscr{V}} u_{et} \varphi_{et} \geq \Delta \tag{8.16c}$$

$$\varphi_{et} \leq y_{et} \qquad e \in \mathscr{E}, t \in \mathscr{V} \tag{8.16d}$$

$$-\varphi_{et} \leq y_{et} + z_{et} \qquad e \in \mathscr{E}, t \in \mathscr{V} \tag{8.16e}$$

$$y_{et} + z_{et} \leq 1 \qquad e \in \mathscr{E}, t \in \mathscr{V} \tag{8.16f}$$

$$y_{et}, z_{et} \in \{0,1\} \qquad e \in \mathscr{E}, t \in \mathscr{V}. \tag{8.16g}$$

Let problem $\mathbf{G}(u^0)$ be feasible for some (fractional) u^0; denote its optimal solution by $(\varphi^*(u^0), \pi^*(u^0), y^*(u^0), z^*(u^0))$, and by $G^*_{u^0}$ the optimal value of the objective function. If $G^* = G^*_{u^0} < 1$, then inequality (8.14) with $\mathscr{I}^1 = \mathscr{J}(y^*(u^0))$ and $\mathscr{I}^0 = \mathscr{J}(z^*(u^0))$ separates u^0, and it does not separate any admissible binary shortest path routing configuration vector $u \in \mathscr{U}$.

As in the case of binary routing vectors, the cut can be made stronger by defining a smaller set $\mathcal{I}^1$. This requires appropriate modification of $\mathbf{G}(u)$ (cf. problem $\mathbf{H}(u)$ in [78]).

In contrast to the LP problem $\mathbf{F}(u)$, problem $\mathbf{G}(u)$ is an MIP. Trying to separate fractional vectors u more effectively, we might consider a linear relaxation of problem $\mathbf{G}(u)$. The definition of $\mathcal{J}(q)$ can be modified to cover fractional vectors q: $\mathcal{J}(q) = \{(e,t) \in \mathcal{E} \times \mathcal{V} : q_{et} > 0\}$, because sets $\mathcal{J}(y^*)$ and $\mathcal{J}(z^*)$ are disjoint. Then, inequality (8.14) is still properly defined in the sense that admissible routing configurations are not separated.

Unfortunately, G_u^* is in general not equal to $S(\mathcal{J}(y^*), \mathcal{J}(z^*); u)$. However, one may still use an approximate separation procedure that consists of solving the linear relaxation and evaluating the resulting value of $S(\mathcal{J}(y^*), \mathcal{J}(z^*); u)$: a valid inequality is found if $S(\mathcal{J}(y^*), \mathcal{J}(z^*); u) < 1$. It should be noted that due to the fact that variables φ_{et} describe circular flows, and constraint (8.12b) requires that negative flows are compensated for by positive flows, in practice usually all nonzero values φ_{et}^* are equal to Δ or $-\Delta$, for some $0 < \Delta \leq 1$. When this is the case, $G_u^* = \Delta \cdot S(\mathcal{J}(y^*), \mathcal{J}(z^*); u)$, and a valid inequality separating u is thus found if $G_u^* < \Delta$.

Thus, we have a set of methods that generate cuts separating a fractional solution vector u: solving problem $\mathbf{H}(u)$ as an MIP, solving problem $\mathbf{G}(u)$ as an MIP, and solving the linear relaxation of problem $\mathbf{G}(u)$. These methods clearly differ with respect to their computational complexity and also the quality of the computed cuts. In practice however, one is not obliged to choose a single method. Instead, one can combine the selected methods into meta-methods by running these methods in sequence or in parallel until the first cut is found or some time limit is reached. This gives rise to different strategies of generating cuts.

Table 8.1 Results of B&C for a six node network

Scenario	After	Strategy	Time [sec.]	Nodes	Cuts
1	-	CPLEX	>4155	>664000*	-
2	-	MIP(G)	>41800*	>28000	>8000
3	-	LR(G)/MIP(G)	>72700*	>66600	>10000
4	-	CC	548	68700	1556
5	-	CC/LR(G)	4081	64000	2485
6	-	CC/MIP(G)	47777	65200	4696
7	-	CC/LR(G)/MIP(G)	13299	24900	2785
8	-	CC/LR(G)/MIP'(G)	4071	14500	1961
9	-	CC/MIP(H)/LR(G)/MIP(G)	34400	43600	3114
10	-	CC/(MIP(H)‖(LR(G)/MIP(G)))	7550	7200	1567
11	4	CPLEX	292	46400	1556
12	9	CPLEX	539	57100	3114
13	10	CPLEX	41	5800	1567

To evaluate the effectiveness of different methods and different strategies of generating cuts, we have performed numerical experiments using a small test network

consisting of six nodes and 28 links with capacities ranging from 24 to 76, and all possible 30 demands. We considered solving the integrated MIP problem of routing and link weights optimization with a B&C method using the CPLEX 10.1 solver. The optimal value of the objective function (the objective was to maximize the minimal residual link capacity) was equal to 6, while the upper bound resulting from the linear relaxation of the problem was equal to 13. The results of the experiments are summarized in Table 8.1. We applied a number of strategies that are defined in column *Strategy*: CPLEX is relying only on standard cuts generated by the solver; CC is generating combinatorial inequalities described in Section 8.4.1; MIP(X) is generating cuts by solving problem X as MIP; LR(X) is generating cuts by solving problem X as LP. Symbols / and ∥ mean that the two methods given as the arguments are run, respectively, in sequence and in parallel. The table provides the information about the total time of the computation, the total number of visited B&C nodes, and the total number of generated user-defined cuts. The asterisk in columns *Nodes* or *Time* means that the computation was aborted due to, respectively, running out of memory or reaching a time limit.

On the one hand, it is evident that there is a need to combine the MIP-based methods of generating general inequalities with the method of generating combinatorial cuts; that is because of the high computational cost of solving problems $\mathbf{H}(u)$ and $\mathbf{G}(u)$ as MIPs, and of the insufficient quality of cuts resulting from solving the linear relaxation of problem $\mathbf{G}(u)$. The results also suggest that the separation procedure based on solving problems $\mathbf{H}(u)$ or $\mathbf{G}(u)$, and as a result the overall B&C approach, do not scale well with the network size, because the integer linear models very quickly become large and hard to solve. On the other hand, however, the quality of cuts that are generated using MIP-based methods is likely to be high. This can be examined by providing the cuts resulting from a particular scenario (cf. scenario 10) as a set of initial user-defined cuts (this fact is indicated in column *After*) and solving the problem again with the CPLEX solver's regular B&C procedure; the quality of generated cuts can be seen from scenarios 11 through 13. Thus, constituting the only exact approach that allows separating fractional solutions, the presented methods seem to be worth investing even more research effort into.

8.5 Heuristic Methods

As already pointed out, shortest path routing problems are NP-hard. Direct formulations are extremely hard to solve, and integer programming approaches can typically resolve only small- to medium-size problems. Moreover, in an operational setting, additional constraints can appear that are difficult to integrate in a mixed-integer programming formulation. Therefore, for large network instances, heuristics can be necessary to find good feasible solutions in limited computing time. Another important application of heuristic methods is that they provide good upper bounds for the branch-and-cut integer programming approach of the form presented in Section 8.3.

8.5.1 Local Search

One of the first heuristic approaches to the shortest path routing problem STEP for its ECMP version was a local search approach developed by Fortz and Thorup [42, 45]. Recently, a similar implementation has been made available in the open source TOTEM toolbox [57].

A solution of the weight setting problem is completely characterized by its vector w of weights. The local search heuristic is based on two different neighborhoods, defined by one of the two following operations applied to w.

Single weight change.   This simple modification consists in changing a single weight in w. We define a neighbor w' of w for each arc $e \in \mathcal{E}$ and for each possible weight $w' \neq w_e$ by setting $w'_e = w'$ and $w'_f = w_f$ for all $f \in \mathcal{E} \setminus \{e\}$.

Evenly balancing flows.   To obtain a good routing *when ECMP is applied*, it is desirable to split the flow as evenly as possible between different arcs.

More precisely, consider a target node t such that some part of the demand going to t goes through a given node u. Intuitively, we would like OSPF routing to split the flow to t going through u evenly along the arcs leaving u. This is the case when every arc in $\delta^+(u)$ belongs to a shortest path from u to t. More precisely, if $\delta^+(u) = \{a_i : 1 \leq i \leq p\}$, and if P_i is one of the shortest paths from the originating node of a_i to t, for $i = 1, 2, \ldots, p$, as illustrated in Figure 8.4, then we want to set w' such that

$$w'_{a_i} + w'(P_i) = w'_{a_j} + w'(P_j) \qquad 1 \leq i, j \leq p,$$

where $w'(P_i)$ denotes the sum of the weights of the arcs belonging to P_i. A simple way of achieving this goal is to set

$$w'(a) = \begin{cases} w^* - w(P_i) & \text{if } a = a_i, \text{ for } i = 1, \ldots, p, \\ w_a & \text{otherwise} \end{cases}$$

where $w^* = 1 + \max_{i=1,2,\ldots,p} \{w(P_i)\}$.

A drawback of this approach is that an arc that does not belong to one of the shortest paths from u to t may already be congested, and the modifications of weights we propose will send more flow on this congested arc, an obviously undesirable feature. We therefore decided to choose at random a threshold ratio θ between 0.25 and 1, and we only modify weights for arcs in the maximal subset B of $\delta^+(u)$ such that

$$\begin{aligned} w_{a_i} + w(P_i) \leq w_{a_j} + w(P_j) \qquad & \forall i : a_i \in B, \ j : a_j \notin B, \\ l^w_a \leq \theta \, c_a \qquad & \forall a \in B, \end{aligned}$$

where l^w_a denotes the load on a resulting from weight vector w. The last relation implies that the utilization of an arc $a \in B$ resulting from the weight vector w is less than or equal to θ, so that we can avoid sending flow on already congested

arcs. In this way, flow leaving u towards t can only change for arcs in B, and choosing θ at random allows us to diversify the search.

This choice of B does not ensure that weights remain below w_{max}. This can be done by adding the condition $\max_{i:a_i \in B} w(P_i) - \min_{i:\,a_i \in B} w(P_i) \leq w_{max}$ when choosing B.
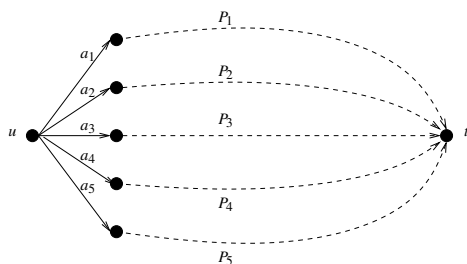
Fig. 8.4 The second type of move tries to make all paths form u to t of equal length

Note that the second type of move does not apply to unsplittable flow routing. In that case, only the first type of move is relevant. Moreover, adapting the algorithm for unsplittable flows requires more work, as solutions with multiple shortest paths must be rejected (or highly penalized in the objective function).

These neighborhoods are embedded in a local search heuristic where cycling is avoided by using hashing tables (this can be seen as a particular tabu search implementation). Some effective diversification schemes have also been proposed. The main strength of this approach is its ability to efficiently recompute the shortest paths and the flows while exploring the neighborhood. As these efficient approaches have also been used in subsequent works, we describe them in Subsection 8.5.3. Recently, Fortz and Ümit [81] managed to significantly improve the results obtained by the heuristic by warm-starting the local search with the dual variables of a multi-commodity flow relaxation of the problem. The idea of using the dual variables as heuristic weights has been concretized in the TOTEM toolbox as of version 3.2 under the name of the *FastIPMetric* module. Klopfenstein and Mamy [56] recently showed that the optimization problem could be made more tractable by restricting the number of possible weight values on arcs to only a few.

One advantage of this approach is that it can be easily extended to take into account multiple demand matrices [43] or multiple scenarios arising, for example, from robustness issues (e.g., link or node failures) [44].

8.5.2 Other Algorithms

Ericsson et al. [37] have proposed a genetic algorithm for the same problem. Solutions are naturally represented as vectors of weights, and the crossover procedure

used is *random keys*, first proposed by Bean [4]. To cross and combine two parent solutions p_1 (elite) and p_2 (non-elite), first generate a random vector r of real numbers between 0 and 1. Let K be a cutoff real number between 0.5 and 1, which will determine if a gene is inherited from p_1 or p_2. A child c is generated as follows: for all genes i, if $r[i] < K$, set $c[i] = p_1[i]$; otherwise, set $c[i] = p_2[i]$. They also implemented a mutation operator that randomly mutates a single weight.

This approach was improved by Buriol et al. [30]. They added a local search procedure after the crossover to improve the population. This hybrid approach, combined with dynamic updates of shortest paths and flows, lead to results competitive in quality to the local search of Fortz and Thorup, with a slightly faster convergence. Another application of genetic algorithms to SPR design can be found in [62].

A simulated annealing approach was proposed by Ben-Ameur [8] for the single path routing case. Another line of heuristic approaches comes from using Lagrangean relaxations of the MIP models (see [9] for single path routing and [51] for ECMP routing).

8.5.3 Effectiveness Issues

To evaluate the cost of a solution represented as a set of weights, we have to compute the shortest paths for all origin-destination pairs, then send the flows along the shortest paths according to the ECMP splitting rule. This could be a bottleneck in the search for good solutions as computing this cost function from scratch is computationally expensive.

There are two basic ways of computing the ECMP flows for a given system of weights w. The first way consists in using an LP formulation. In such a formulation a regular weight system w (for the notion of regularity see Section 8.2.1) is given, and the unknowns are the accumulated link flows x_{et} (recall that x_{et} is the total flow to destination node t on link e). As explained in [72], such a formulation can be obtained as a linear program resulting from a subset of the complete problem formulation presented in Section 8.7.1. In the formulation, variables w_e are fixed to the values of the current weights, routing variables u_{et} are made continuous, capacity constraints (8.18d) are skipped, and the objective is changed to maximizing the following function:

$$F = (|\mathcal{V}| \cdot |\mathcal{E}|) \textstyle\sum_{s \in \mathcal{V}} \sum_{s \in \mathcal{V} \setminus \{v\}} r_{st} + \sum_{e \in \mathcal{E}} \sum_{t \in \mathcal{V} \setminus \{a(e)\}} (1 - u_{et}). \tag{8.17}$$

However, for heuristic solutions, we can also apply a fast algorithmic approach to compute the flows. This can be done using a two-step algorithm based on the shortest path computation. In the first step we compute all the w-shortest paths for all node pairs, and then, in the second step, we recursively assign flows to the paths computed in the first phase (see [45] or Algorithm 7.1 in [68]).

In most heuristic approaches, the number of changes in the shortest paths graph and in the flows is very small between neighboring solutions. Hence, using fast

updates of shortest paths and flows is crucial to make heuristics effective. We now briefly review these approaches.

With respect to shortest paths, this idea is already well studied [74], and we can apply their algorithm directly. Their basic result is that, for the recomputation, we only spend time proportional to the number of arcs incident to nodes s whose distance r_{st} to t changes. In typical experiments there were only very few changes, so the gain is substantial – in the order of factor 15 for a 100 node graph. An improved algorithm was recently proposed by Buriol et al [31].

To update the flows, a similar approach, described in [45], can be used. Experiments reported in that paper show that using dynamic updates of shortest paths and flows make the algorithm from five to 25 times faster, with an average of 15 times faster.

8.6 Numerical Results

In this section, we present selected numerical results obtained with the two presented optimization approaches. The exact integer programming approach described in Section 8.3 is illustrated in Subsection 8.6.1 for the case of unsplittable shortest path routing . In Subsection 8.6.2, we then present results for one of the local search heuristics discussed in Section 8.5.

8.6.1 Integer Programming Approach

Several variants of the two-phase integer programming approach have been implemented as part of the network optimization library DISCNET [2]. This implementation is especially designed for the unsplittable shortest path routing version and uses only binary link-flow variables (or, alternatively, binary path variables) instead of the SP tree variables and the aggregated flow variables to model the routing in the master problem formulation. The corresponding conflict constraints for the unsplittable shortest path routing version are separated via combinatorial heuristics or, if these fail, via the client problem analogously to the conflict inequalities for the ECMP routing version, as described in Sections 8.3 and 8.4. In addition to these inequalities, which describe the admissible routing patterns independently of traffic demands and link capacities, the DISCNET implementation also uses cutting planes based on the induced traffic flows and the link capacities. In practice, induced cover inequalities based on the precedence-constrained knapsacks defined by a single link capacity constraint and the subpath consistency among the paths across that link proved to be very useful [13, 18]. All data structures and algorithms are implemented in C++ using the library LEDA 4.1 [1]; the linear programs arising in the solution process are solved with CPLEX 11.0 [53]. Further details, including a description of all cutting planes and separation algorithms used, of the especially

Table 8.2 Integer programming results for unsplittable shortest path routing

Problem	Nodes	Links	Demands	LP	LB	Sol	Nodes	Gap (%)	Time (s)
Atlanta	15	22	210	0.65	0.86	0.86	30	0.0	10.3
Dfn-bwin	10	45	90	0.34	0.69	0.69	89	0.0	26.5
Dfn-gwin	11	21	110	0.50	0.51	0.51	521	0.0	16.3
Di-yuan	11	42	22	0.25	0.62	0.62	33	0.0	1.8
France	25	45	300	0.60	0.71	0.74	76	5.0	10000.0
Germany50	50	88	662	0.64	0.64	0.73	56	12.7	10000.0
NewYork	16	49	240	0.44	0.62	0.62	15	0.0	54.9
Nobel-EU	28	41	378	0.44	0.44	0.45	75	0.3	10000
Nobel-GER	17	26	121	0.64	0.73	0.73	101	0.0	114.1
Nobel-US	14	21	91	0.48	0.49	0.49	77	0.0	20.4
Norway	27	51	702	0.54	0.54	0.62	99	14.9	10000.0
PDH	11	34	24	0.34	0.80	0.80	85	0.0	6.37
Pioro40	40	89	780	0.38	0.38	0.45	311	19.6	10000.0
Polska	12	18	66	0.82	0.93	0.93	2149	0.0	200.2
Sun	27	102	67	0.29	0.39	0.70	102	76.8	10000.0
TA1	24	55	396	0.30	0.93	0.93	11	0.0	289.2

tailored branching schemes, and of the problem-specific primal heuristics, can be found in [13] and [14].

Table 8.2 shows computational results for a collection of benchmark problems taken from the Survivable Network Design Data Library [66]. All computations were performed on a Linux 2.6 machine with an Intel Core2 CPU running at 2.66 GHz and with 4 GByte RAM. The two-phase decomposition algorithm was run with a total CPU time limit of 10,000 seconds on each problem instance.

The underlying networks are bidirectional and have the same capacity for both directions of all links. The number of nodes, bidirected links, and nonzero traffic demands is shown in the first columns of Table 8.2. Column LP shows the lower bound obtained by solving the linear relaxation of (8.3) at the root node of the master problem's branch-and-bound tree. The columns LB and Sol show the best lower bound proved and the value of the best solution found by the two-phase decomposition algorithm within the given time limit. The remaining columns show the number of explored branch-and-bound nodes, the residual optimality gap, and the total CPU time until either optimality was proved or the time limit exceeded.

We observe that small- and medium-size instances can be solved optimally. For large problems optimality cannot always be achieved. Instances with dense networks that have lots of short potential routing paths for most demand pairs are more difficult than those where the underlying networks are fairly sparse. For instances with dense networks, lots of violated conflict constraints are separated during the execution of the algorithms, which often drastically slows down the solution of the linear relaxation. For the most difficult instances, only few branch-and-bound nodes could be explored.

Figure 8.5 illustrates the importance of optimizing the routing weights in practice. It shows the different link loads that would result with unsplittable shortest path routing from three commonly used default weight settings, and those resulting

(a) Inverse link capacities (perturbed)

(b) Unit lengths (perturbed)

(c) Geographic lengths

(d) Optimized routing lengths
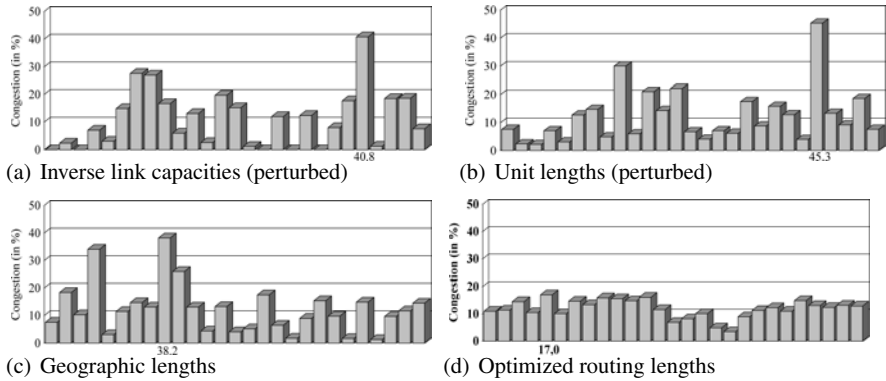
Fig. 8.5 Link congestion values in G-WiN for several routing metrics

from the optimized routing weights in the German national research and education network G-WiN with capacities and traffic demands of August 2001. Even without using the traffic splitting possibilities of the ECMP routing version, the traffic is distributed much more evenly for the optimized metric. The peak congestion is not even half of that for the default settings, which significantly reduces packet delays and loss rates and improves the network's robustness against unforeseen traffic changes and failures.

8.6.2 Heuristic Methods

In this section, we present results obtained with the TOTEM implementation, called IGP-WO, of the local search heuristic [57] described in Section 8.5.

The instances used are the same as in Section 8.6.1, but with ECMP traffic splitting allowed. The heuristic tries to optimize the normalized cost function defined in [43] (see also Section 8.7.2). As a side product, this cost function also maintains small maximum utilization.

As proposed in [81], the heuristic is started with weights corresponding to the values of the dual variables of a multi-commodity flow relaxation of the problem. This relaxation also provides a lower bound on the optimal value. 100 iterations of the local search heuristic are performed.

Results are presented in Table 8.3, where they are compared based on the maximum utilization and the normalized cost function from [43]. MCNF is the lower bound obtained with the multi-commodity flow relaxation of the problem, Inv-cap is obtained with weights inversely proportional to the link capacities (the standard used by most operators), and Dual Values is the solution obtained by setting weights using dual values from the MCNF relaxation. We report max-utilization for three flavors of IGP-WO, applied starting with the dual values solution. NC is the solution after

100 iterations of local search with the normalized cost function as objective, MU-H after 100 iterations with max-utilization as objective, and MU-T after 1,000 iterations with max-utilization as objective. Note that performing 100 iterations takes less than five seconds on a standard desktop PC, which makes the heuristic approach very attractive if many scenarios have to be considered, or if operators need to react quickly to a sudden event.

Table 8.3 Heuristic results for ECMP routing

| Problem | Maximum Utilization | | | | | | Normalized Cost | | | |
| | MCNF | Inv-cap | Dual Values | IGP-WO | | | MCNF | Inv-cap | Dual Values | IGP-WO |
				NC	MU-H	MU-T				
Atlanta	0.65	0.95	0.96	0.92	0.83	0.74	0.14	0.20	0.28	0.20
Dfn-bwin	0.34	0.42	0.51	0.69	0.42	0.42	0.11	0.11	0.12	0.11
Dfn-gwin	0.50	0.67	0.83	0.56	0.52	0.50	0.11	0.11	0.13	0.11
Di-yuan	0.25	0.77	0.75	0.62	0.75	0.50	0.10	0.14	0.17	0.11
France	0.60	1.64	1.62	0.79	0.75	0.65	0.11	23.87	25.00	0.15
Germany50	0.64	1.76	1.56	0.75	0.70	0.65	0.08	8.78	19.28	0.13
NewYork	0.44	0.85	0.95	0.69	0.84	0.64	0.11	0.14	0.18	0.13
Nobel-EU	0.44	0.77	0.77	0.59	0.44	0.44	0.08	0.11	0.11	0.10
Nobel-GER	0.64	1.45	1.21	0.67	0.64	0.64	0.11	12.31	6.43	0.13
Nobel-US	0.48	0.66	0.79	0.56	0.50	0.49	0.10	0.12	0.14	0.12
Norway	0.54	0.90	1.08	0.69	0.74	0.60	0.12	0.15	0.29	0.13
PDH	0.34	1.10	3.32	0.80	0.51	0.51	0.09	1.01	124.66	0.14
Pioro40	0.38	0.71	0.56	0.48	0.40	0.38	0.09	0.10	0.10	0.09
Polska	0.82	1.04	1.08	0.90	0.91	0.87	0.25	0.53	0.54	0.25
Sun	0.29	2.04	1.89	0.67	0.71	0.34	0.09	65.06	27.99	0.13
TA1	0.33	0.89	0.76	0.66	0.35	0.32	0.09	0.14	0.13	0.10

We observe that Inv-cap and Dual Values are often very far from the lower bound, and that IGP-WO improves this solution substantially, both for max-utilization and the normal cost. Optimization with max-utilization as objective function performs much better with that criterion, but at the price of a very local improvement in the routing. For instances France, Germany50, Nobel-GER, PDH, and Sun, the improvement is really impressive as the heuristic is able to decrease the maximum utilization and the normalized cost below 1, while simple heuristics lead to an over-congested network.

Note that for most instances, the dual values perform worse than Inv-Cap. Nevertheless, we observed that in practice, this solution is a better starting point for the local search heuristic as the structure of the routing is closer to an optimal one despite the worst objective value.

8.7 Selected Extensions

In this section we briefly extend the presentation of the previous sections to cover several important issues not discussed so far. In particular, we present a full mixed-integer programming formulation of STEP (Section 8.7.1), discuss additional routing constraints and various objective functions (Section 8.7.2), show how resource dimensioning can be taken into account (Section 8.7.3), and show how optimization of resilient weight systems can be approached (Section 8.7.4). Finally, in Section 8.8, we give historical remarks.

8.7.1 General MIP Formulation

For completeness, a full mixed-integer programming formulation of the basic SPR problem (STEP) considered earlier in this chapter is given below. The formulation is explained in detail, so the knowledge of Section 8.2 is sufficient for its understanding.

General ECMP Traffic Engineering Problem

find

x_{et} ECMP flow destined to node t induced by system w on link e

z_{vt} common value of the ECMP flow destined to node t, assigned to the links outgoing from node v and belonging to the shortest paths from v to t

u_{et} variable indicating whether link e is on a shortest path to node t

r_{st} length of the w-shortest path from s to t

w_e weight assigned to link e

Z maximum over link utilization factors

minimize

$$Z \qquad\qquad\qquad (8.18a)$$

subject to

$$\sum_{e\in\delta^-(t)}x_{et} = D_t \qquad\qquad t\in\mathcal{V} \qquad\qquad (8.18b)$$

$$\sum_{e\in\delta^+(v)}x_{et} - \sum_{e\in\delta^-(v)}x_{et} = d_{vt} \qquad\qquad t\in\mathcal{V},\, v\in\mathcal{V}\setminus\{t\} \qquad (8.18c)$$

$$\sum_{t\in\mathcal{V}\setminus\{a(e)\}}x_{et} \le Zc_e \qquad\qquad e\in\mathcal{E} \qquad\qquad (8.18d)$$

$$0\le x_{et}\le D_t u_{et} \qquad\qquad e\in\mathcal{E},\, t\in\mathcal{V}\setminus\{a(e)\} \quad (8.18e)$$

$$0\le z_{a(e)t} - x_{et}\le D_t(1-u_{et}) \qquad\qquad e\in\mathcal{E},\, t\in\mathcal{V}\setminus\{a(e)\} \quad (8.18f)$$

$$1 - u_{et}\le w_e + r_{b(e)t} - r_{a(e)t} \qquad\qquad e\in\mathcal{E},\, t\in\mathcal{V}\setminus\{a(e)\} \quad (8.18g)$$

$$w_e + r_{b(e)t} - r_{a(e)t}\le M(1-u_{et}) \qquad\qquad e\in\mathcal{E},\, t\in\mathcal{V}\setminus\{a(e)\} \quad (8.18h)$$

$$r_{st}\ge 1 \qquad\qquad (s,t)\in\mathcal{V}^{|2|} \qquad\qquad (8.18i)$$

$$r_{vv} = 0 \qquad\qquad v\in\mathcal{V} \qquad\qquad (8.18j)$$

$$u_{et}\in\{0,1\} \qquad\qquad e\in\mathcal{E},\, t\in\mathcal{V}\setminus\{a(e)\} \quad (8.18k)$$

$$w_e\in\{1,2,\dots,K\} \qquad\qquad e\in\mathcal{E} \qquad\qquad (8.18l)$$

$$x,z,r,Z\ge 0 \qquad\qquad\qquad\qquad (8.18m)$$

$$x,z,r,Z\in\mathbb{R} \qquad\qquad\qquad\qquad (8.18n)$$

Above, M is a constant ("big-M") not less than the difference in length of any two paths in the network graph. For example, $M = K\cdot|\mathcal{E}|$ would suffice. Still, for computing lower bounds in the linear relaxations of problem (8.18) it is advantageous to use an M as small as possible, so this constant can be made dependent on the pairs of nodes (s,t), and thus potentially reduced for some node pairs.

Subproblem (8.18a)–(8.18d) minimizes the maximum link utilization factor Z and is an aggregated node-link formulation of a capacitated multi-commodity flow allocation problem with aggregated flows x, where x_{et} denotes the total traffic destined to node t carried on link e. The next two constraints express the ECMP rule of traffic routing by means of binary routing variables u. Each variable u_{et} is supposed to be equal to 1 if, and only if, link e belongs to at least one shortest path (with respect to weight system w) from its originating node $a(e)$ to node t. Constraint (8.18e) forces traffic destined to node t to use only the links allowed by the routing configuration u (i.e., links $e\in\mathcal{E}$ with $u_{et}=1$), while constraint (8.18f) ensures that in each node the traffic to destination node t is split equally among the links assigned to that destination. For node $v\in\mathcal{V}$ and destination $t\in\mathcal{V}$ this common value of equal split is expressed by variable z_{vt}. Finally, the shortest path routing constraints (8.18g)–(8.18j) ensure that the routing vector u defines shortest paths consistent with the weight system w. Each variable r_{vt} is supposed to express the distance (length of the shortest path with respect to w) from node v to node t. The quantity $q_e := w_e + r_{b(e)t} - r_{a(e)t}$ measures the difference between the length of the shortest path which starts in node $a(e)$, goes over link e, and ends in node t, and the distance from the starting node of e to t. Thus, link e is on a shortest path to node t if, and only if, $w_e + r_{b(e)t} - r_{a(e)t} = 0$.

The correctness of formulation (8.18) follows easily from the discussion of the independent STEP master and ISP client models in Section 8.3.

Note that in formulation (8.18) the link weights can be either integer (as assumed) or continuous in interval $[1, K]$. In the latter case the regularity of the weight system resulting from (8.18) is assured by constraint (8.18g).

We would like to finally recall what has already been stated on several occasions, that because of the intrinsic difficulty of STEP, such formulations as (8.18) are virtually impossible to solve to optimality by contemporary integer programming solvers for medium-size network instances.

8.7.2 Additional Routing Constraints and Other Objective Functions

The general STEP formulation (8.18) can be adjusted to a version with a limited split of flows at the nodes, which can be useful in practice. This is done by adding the following constraint to formulation (8.18):

$$\sum_{e \in \delta^+(v)} u_{et} \leq n \qquad\qquad v \in \mathcal{V}, \, t \in \mathcal{V} \setminus \{v\}. \tag{8.19}$$

This constraint forces at most n nonzero flows allowed to each destination t at each node $v \in \mathcal{V}$. Certainly, if we put $n = 1$, then we will force unsplittable shortest path routing, admitting only single shortest paths.

The same adjustment can be applied to the STEP formulation (8.3) introduced in Section 8.3.1. Adding constraint (8.19) with $n = 1$ and removing variables z and constraints (8.3e) from formulation (8.3) will result in an integer progamming formulation for the STEP master problem for unsplittable shortest path routing.

There can also be other objective functions used in formulation (8.18). A simple variant of such a function is the maximization of the minimum unused link capacity, denoted by Z, with the following constraint in place of constraint (8.18d): $\sum_{t \in \mathcal{V} \setminus \{a(e)\}} x_{et} + Z \leq c_e$, for all links $e \in \mathcal{E}$. Note that with this objective the problem is always feasible, provided we allow solutions with $Z < 0$. However, a true feasible routing solution is obtained when the resulting objective value is nonnegative.

Another important objective is obtained by using the Kleinrock delay function [47], to be minimized: $F = \sum_{e \in \mathcal{E}} \frac{y_e}{c_e - y_e}$, where y_e denotes the load of link e, $y_e = \sum_{t \in \mathcal{V} \setminus \{a(e)\}} x_{et}$. As with the previous objectives, the Kleinrock function helps us avoid congestion by penalizing heavily loaded links. Observe that this objective function is convex, and hence can be approximated with a piecewise linear function. This leads to linear relaxations of the corresponding convex problem, as demonstrated in [45] (see also [68]).

As discussed in [43], an issue in these cost function formulations is that they do not provide a universal measure of congestion. It is natural to require that the maximum utilization remain below 1, independently of the network topology and demand matrix. Similarly, we would like a universal cutoff value for the cost function, independently of the network topology and demand matrix. Fortz and Thorup defined in [43] a normalized version of the cost function, for which it is natural to say that a routing solution congests a network if the cost is greater than 1.

A comparison between problems with several objective functions can be found in [50].

8.7.3 Resource Dimensioning

So far we have been investigating versions of STEP with fixed link capacities c_e, $e \in \mathcal{E}$. Still, it can be also important to combine SPR routing optimization with simultaneous dimensioning of the network resources, such as link and node capacities. For example, design of an overlay IP network can typically consist in designing IP link capacities (which will be then leased from the carrier network) and an SPR weight system to be used to route packets in the network of leased links. The objective of such a design problem, in addition to keeping link loads reasonable, can be minimizing the capacity, and hence the cost of the leased links.

Certainly, STEP formulation (8.18) can be extended to cover the new situation. For that, link capacities c_e, $e \in \mathcal{E}$, are made variable, and denoted by y_e, $e \in \mathcal{E}$, objective Z can be moved to constraints ($Z \leq 0.8$, for example), and the total link cost $\sum_{e \in \mathcal{E}} \xi_e y_e$ can be used in (8.18a) as the objective to be minimized (assuming certain link unit costs ξ_e, $e \in \mathcal{E}$, for example, the costs of leasing the link capacity from the carrier).

Moreover, modularity of link capacity can be taken into account assuming integer values for variables y_e, $e \in \mathcal{E}$, and the resulting link capacity equal to $m \cdot y_e$, for some fixed module m expressed in Mbit/s. As usual, considering modularity adds difficulty to the problem. Network design problems for SPR of this kind are discussed in [9, 13, 16, 18, 48, 68].

8.7.4 Resilient Routing

Network elements such as links and nodes are subject to failures, and hence the resilience issue should be taken into account when designing a weight system w. In principle, the weight system used in an autonomous system (domain) could be re-optimized when a failure occurs and then sent out to the routers in order to modify the packet forwarding tables so that at least the failed links are excluded from the routing pattern. In practice, such dynamic weight updates are difficult and even risky to implement. Therefore, another approach to weight resiliency is usually considered, where the link weights are only modified by assuming infinite weight for the failing links. This idea and resulting optimization issues are summarized below.

Let $s \in \mathscr{S}$ denote a failure state, where $\mathscr{S}$ is the family of all the considered failure states (for example, all single link failures), and each failure state is determined by the set of failing links, i.e., $s \subset \mathcal{E}$. Now, if failure s occurs, the routers are notified about the set s of failing links, and they locally modify the current weight system

w to obtain a new system w', where $w'_e = w_e$, $e \in \mathscr{E} \setminus s$, and $w'_e = +\infty$, $e \in s$, and recompute the forwarding tables accordingly.

This weight modification strategy can be taken into account at the weight system design stage, for example by an appropriate modification of STEP and its resolution methods. In fact, it is a straightforward exercise to write down a modification of problem (8.18) that, for example, minimizes objective Z over all non-failing links in the normal state and in all the failure states from the assumed failure scenario $\mathscr{S}$ [72]. Additionally, in such a formulation we can directly take into account that demand volumes can be made different in different failure states. Certainly, these modifications add difficulty to the considered problem (which already in its basic version is very difficult) and to possible integer programming and heuristic approaches.

In heuristic approaches the need of computing values of the objective function for each failure state $s \in \mathscr{S}$ makes the evaluation of a single weight system very expensive. To cope with this issue, Fortz and Thorup [44] adapted their local search heuristic (see Section 8.5.1) by considering only a critical set of failure states, representative for the whole failure scenario $\mathscr{S}$. This allows us to achieve resilient weight systems while keeping the problem manageable in size.

Certainly, resilient routing design can be combined with resource (link) dimensioning; see, for example, [48]. More on design of resilient SPR routing can be found in [13, 16, 18, 25, 35, 59, 64].

8.8 Historical and Literature Notes

Practical relevance of STEP and related shortest path routing optimization problems has attracted attention of many researchers in operations research and telecommunications during the last decade. Below, we give a historical survey of the work on the SPR optimization issues.

Ben-Ameur and Gourdin [5, 6], Broström and Holmberg [21, 22, 27, 28], and Bley [13] studied the properties of path sets that correspond to (undirected) shortest path routing patterns, i.e., systems that are generated by weight systems. In particular, they revealed different necessary conditions for such sets to have a compatible weight system. Broström and Holmberg considered path sets arising from the ECMP shortest multi-path routing, while Ben-Ameur, Gourdin, and Bley were focusing on the unsplittable shortest path routing case. Analyzing the linear model (in particular its dual) for finding a weight system inducing a given set of paths, Broström and Holmberg [23, 24] have proposed strong compatibility conditions and polynomial algorithms to check whether these conditions are satisfied or not. Recently, also Tomaszewski, Pióro et al. studied these issues in [78, 79]. A complete combinatorial description of all valid paths sets, however, is not known. Bley [13] proved that the inclusion-wise minimal invalid path sets for unsplittable shortest path routing can become arbitrarily large and that it is NP-hard to find the smallest conflict in a given path set.

The inverse shortest paths problem (ISP) was considered by Ben-Ameur and Gourdin [5, 6], who devised linear and integer programming models for finding a weight system that induces a prescribed set of shortest paths (or proves that no such weights exist). Also Broström and Holmberg [22, 28] considered mixed-integer linear programming formulations to address the ISP problem. Farago et al. [38, 39] studied a special case of ISP where the given paths are to be the shortest paths with respect to the number of links (hops), and the task is to find the link weights so that all these paths are unique shortest paths. Bley [12] showed that finding a compatible integer weight system is computationally hard if the range of admissible weights is bounded.

Several groups studied problems similar to ISP in the context of data reengineering, where the task is to find weights that resemble (as closely as possible) the known distances or the presumed shortest paths and deviate as little as possible from a given set of reference link weights. Typically, it is not necessary to exactly match the presumed shortest paths or distances, and the integrality of the weights is also not an issue in this context. Burton and Toint [32, 33] solved this problem as a continuous convex quadratic programming problem, while Tong and Lam [80] proposed a conjugate gradient method for its solution. The computational complexity of this variant of the inverse shortest paths problem was analyzed by Fekete et al. [40].

As far as the complexity of STEP is concerned, Jüttner et al. [54] showed that it is NP-hard to find a shortest multi-path routing pattern that splits traffic according to the ECMP rule and whose induced traffic flows do not exceed some given fixed link capacities. Fortz and Thorup [45] proved that the minimum congestion that can be obtained with such a routing pattern cannot be polynomially approximated within a factor less than $3/2$, unless $\mathsf{P} = \mathsf{NP}$. Bley [11] proved that, for unsplittable shortest path routing, the problem is inapproximable within a factor of $\Omega(|\mathscr{V}|^{1-\varepsilon})$ for any $\varepsilon > 0$. In [15], Bley also proposed several polynomial-time approximation algorithms for the unsplittable shortest path routing version of STEP and studied the approximability of the fixed charge and capacitated network design problem with unsplittable shortest path routing.

One of the first mixed-integer programming formulations of the shortest path routing problem was presented by Bley et al. [16], who considered a (survivable) network design problem with unsplittable shortest path routing. This formulation contains binary link routing variables for end-to-end routing paths, integer variables for the link weights, and continuous variables for the distances between the node pairs induced by the weights. The interdependencies between the routing path variables, weights, and the distance variables are expressed analogously as in formulation (8.18), using a quadratic number of linear constraints with big-M coefficients for the binary arc routing variables. The uniqueness of the shortest paths is guaranteed in a quite specific way by adding a predetermined perturbation to the integer arc length variables.

Formulation (8.18) for the ECMP version (which works for unique shortest paths as well) of STEP is due to Tomaszewski and was first published in [69] (see also [70], and Chapter 7 in monograph [68] by Pióro and Medhi). Another mixed-integer programming formulation of this kind was independently proposed by Holmberg

and Yuan [52]. Recently, Parmar et al. [67] presented an integer programming approach for STEP using a formulation analogous to (8.18).

Among the first attempts to formulate shortest path routing problems as integer linear programs without variables for the routing weights were those made by Staehle et al. [77] and Milbrandt [60]. They proposed mixed-integer programming models with binary link-flow variables for finding an unsplittable shortest path routing pattern that minimizes a linear combination of the maximum and the average link congestion. These models ensure that the computed routing paths satisfy the subpath consistency, but they still admit invalid routing patterns as integer solutions.

To our knowledge, the first mathematically correct mixed-integer programming approach without additional variables for the routing weights was presented by Bley and Koch [18] for a survivable network design problem with unsplittable shortest path routing. In their model the only binary variables, besides the link capacity variables, are link routing variables. In order to ensure that a valid unsplittable shortest path routing pattern is obtained, they separate inequalities of type (8.3f) to cutoff invalid routing patterns, using the basic approach presented in Section 8.3.

Since then, numerous solution approaches based on the decomposition idea discussed in Section 8.3 have been presented for different variants of shortest path routing problems by Bley [13], Bourquia et al. [20], Broström and Holmberg [23, 26] (network design problem with ECMP based on SP graphs modeled with binary variables and with subpath consistency ensured by binary variables indicating whether or not a node is reachable from another node in an SP graph; it is solved with a Lagrangean heuristic, where feasible solutions are generated using a tabu search method), Giovanni et al. [48] (minimum cost resilient fixed-charge network design problem with ECMP; fixed link weights), Pióro et al. [69, 70] (unique shortest paths, meta-heuristics assuring subpath consistency applied in the first phase), Eremin et al. [36] (k-splittable routing), and Tomaszewski et al. [78, 79]. Holmberg and Yuan [52] generalized the decomposition approach to solve a network design problem with unsplittable shortest path routing and multicast commodities. This case was also considered by Prytz [73].

In fact, shortest path routing problems have been more frequently tackled with heuristic methods than with integer programming methods. This is understandable, since STEP, as discussed in this chapter, is very difficult to solve with mixed-integer programming techniques.

A specific simple approach to finding a reasonable weight system was proposed by Pióro et al. [69, 70, 76] and independently by Wang et al. [82]. It considers a linear fractional multi-commodity flow allocation problem (with minimizing the cost of routing as the objective), and uses the resulting optimal dual variables as the link weights. A similar approach was also used by Fortz and Ümit [81] as a good starting solution for a local search approach.

Lagrangian relaxation techniques, local search algorithms, and meta-heuristics, such as simulated annealing, genetic algorithms, and tabu search, which all use the link administrative weights as primary decision variables, are conceptually much easier, and have become quite common to approximately solve shortest path routing optimization problems. It seems that the first successful method to solve the problem

using meta-heuristics was proposed by Fortz and Thorup [42, 45]. It was able to find efficient weight systems for large-size networks in reasonable computing time (see Section 8.5 for details). Other relevant heuristic methods are described in [8, 9, 16, 23, 30, 34, 37, 44, 46, 49, 51, 55, 58, 62, 68, 69].

8.9 Concluding Remarks

The chapter summarizes over a decade research in optimizing weight systems in the Internet Protocol networks. We have concentrated on a representative problem referred to as STEP – an optimization problem related to intra-domain traffic engineering in networks based on shortest path routing protocols like OSPF or IS-IS. Due to great practical relevance, problems of this type have been intensively studied in the literature. The presented chapter is an attempt to summarize the existing knowledge with emphasis on the developments achieved by the chapter authors in their previous work.

We have discussed modeling and resolutions issues of STEP, using the language and methods of multi-commodity networks, eventually based on integer programming. We have described up-to-date exact and heuristic approaches to STEP, and summarized the theoretical results underlying the approaches.

We hope that the presented material forms a solid base for both experienced and less-experienced researchers for getting familiar with the field and for adding their own contributions to it.

Finally, the authors wish to express their gratitude to the anonymous reviewer whose remarks based on deep understanding of the shortest path routing helped to improve the final version of this chapter.

References

1. Algorithmic Solutions Software GmbH, Schuetzenstrasse 3–5, D-66123 Saarbruecken, Germany: LEDA—Library of Efficient Data types and Algorithms (2000–2007). URL http://www.algorithmic-solutions.com/leda
2. atesio GmbH, Sophie-Taeuber-Arp-Weg 27, D-12205 Berlin, Germany: DISCNET – Network optimization software library (2000–2007). URL http://www.atesio.de
3. Awduche, D.: MPLS and traffic engineering in IP networks. IEEE Communications Magazine **37**, 42–47 (1999)
4. Bean, J.: Genetic algorithms and random keys for sequencing and optimization. ORSA J. on Computing **6**, 154–160 (1994)
5. Ben-Ameur, W.: Multi-hour design of survivable classical IP networks. International Journal of Communication Systems **15**, 553–572 (2002)
6. Ben-Ameur, W., Gourdin, E.: Internet routing and related topology issues. SIAM Journal on Discrete Mathematics **17**(1), 18–49 (2003)
7. Ben-Ameur, W., Gourdin, E., Liau, B., Michel, N.: Dimensioning of Internet networks. In: Proceedings of the 2nd International Workshop on the Design of Reliable Communication Networks (DRCN 2000), Munich, Germany (2000)
8. Ben-Ameur, W., Gourdin, E., Liau, B., Michel, N.: Optimizing administrative weights for efficient single-path routing. In: Proceedings of Networks 2000 (2000)
9. Bley, A.: A Lagrangian approach for integrated network design and routing in IP networks. In: Proceedings of the 1st International Network Optimization Conference (INOC 2003), Paris, France, pp. 107–113 (2003)
10. Bley, A.: Finding small administrative lengths for shortest path routing. In: Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal, vol. 1, pp. 121–128 (2005)
11. Bley, A.: On the approximability of the minimum congestion unsplittable shortest path routing problem. In: Proceedings of the 11th Conference on Integer Programming and Combinatorial Optimization (IPCO 2005), Berlin, Germany, pp. 97–110 (2005)
12. Bley, A.: Inapproximability results for the inverse shortest paths problem with integer lengths and unique shortest paths. Networks **50**(1), 29–36 (2007)
13. Bley, A.: Routing and capacity optimization for IP networks. Ph.D. thesis, Technische Univertität Berlin (2007)
14. Bley, A.: An integer programming algorithm for routing optimization in IP networks. In: Proceedings of the 16th Annual European Symposium on Algorithms (ESA 2008), Karlsruhe, Germany, pp. 198–209 (2008)
15. Bley, A.: Approximability of unsplittable shortest path routing problems. Networks **54**(1), 23–46 (2009)
16. Bley, A., Grötschel, M., Wessäly, R.: Design of broadband virtual private networks: Model and heuristics for the B-WiN. In: N. Dean, D. Hsu, R. Ravi (eds.) Robust Communication Networks: Interconnection and Survivability, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 53, pp. 1–16. American Mathematical Society (1998)
17. Bley, A., Koch, T.: Optimierung des G-WiN. DFN-Mitteilungen **54**, 13–15 (2000)
18. Bley, A., Koch, T.: Integer programming approaches to access and backbone IP-network planning. In: Modeling, Simulation and Optimization of Complex Processes: Proceedings of the 3rd International Conference on High Performance Scientific Computing, pp. 87–110. Hanoi, Vietnam (2006)

19. Bley, A., Pattloch, M.: Modellierung und Optimierung der X-WiN Plattform.   DFN-Mitteilungen **67**, 4–7 (2005)
20. Bourquia, N., Ben-Ameur, W., Gourdin, E., Tolla, P.: Optimal shortest path routing for Internet networks. In: Proceedings of the 1st International Network Optimization Conference (INOC 2003), Paris, France, pp. 119–125 (2003)
21. Broström, P.: Optimization models and methods for telecommunication networks using OSPF. PhD dissertation, Linköping University, Sweden (2006). Linköping Studies in Science and Technology. Dissertation no. 1032
22. Broström, P., Holmberg, K.: Determining the non-existence of compatibel OSPF weights. In: D. Yuan (ed.) Nordic MPS 2004, no. 14 in Linköping Electronic Conference Proceedings, pp. 7–21. Linköping University Electronic Press (2004)
23. Broström, P., Holmberg, K.: Design of IP/OSPF networks using a Lagrangean heuristic on an in-graph based model. In: L. Gouveia, C. Mourao (eds.) INOC 2005, pp. 702–709. University of Lisbon, Lisbon, Portugal (2005)
24. Broström, P., Holmberg, K.: Determining the non-existance of a compatible OSPF metric. In: L. Gouveia, C. Mourao (eds.) INOC 2005, pp. 702–709. University of Lisbon, Lisbon, Portugal (2005)
25. Broström, P., Holmberg, K.: Multiobjective design of survivable IP networks. Annals of Operations Research **147** (2006)
26. Broström, P., Holmberg, K.: Design of OSPF networks using subpath consistent routing patterns. Research Report LiTH-MAT-R-2007-05, Department of Mathematics, Linköping Institute of Technology, Sweden (2007). Under revision for publication.
27. Broström, P., Holmberg, K.: On the extremal structure of an OSPF related cone. Vietnam Journal on Mathematics **35**(4), 507–522 (2007)
28. Broström, P., Holmberg, K.: Valid cycles: A source of infeasibility in OSPF routing. Networks **52**(4), 206–215 (2008)
29. Broström, P., Holmberg, K.: Compatible weights and valid cycles in non-spanning OSPF routing patterns. Algorithmic Operations Research **4**, 19–35 (2009)
30. Buriol, L. S., Resende, M. G. C., Ribeiro, C. C., Thorup, M.: A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. Networks **46**(1), 36–56 (2005)
31. Buriol, L. S., Resende, M. G. C., Thorup, M.: Speeding Up Dynamic Shortest-Path Algorithms. INFORMS Journal On Computing **20**(2), 191–204 (2008)
32. Burton, D.: On the inverse shortest path problem. Ph.D. thesis, Department of Mathematics, Facultés Universitaires ND de la Paix, Namur, Belgium (1993)
33. Burton, D., Toint, P.: On an instance of the inverse shortest paths problem. Mathematical Programming **53**, 45–61 (1992)
34. Dzida, M., Petterson, M., Duelli, M., Zagożdżon, M., Pióro, M., Menth, M., Żotkiewicz, M.: Three methods for optimizing single-shortest path routing. In: NGI 2008 Conference on Next Generation Internet Networks, Cracow, Poland (2008)
35. Dzida, M., Zagożdżon, M., Pióro, M.: Optimization of resilient weight systems for shortest-path routing in IP networks. In: 6th International Workshop on Design and Reliable Communication Networks. La Rochelle, France (2007)
36. Eremin, A., Ajili, F., Rodosek, R.: A set-based approach to the optimal IGP weight setting problem. In: Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal, vol. 1, pp. 386–392 (2005)
37. Ericsson, M., Resende, M. G. C., P. M., P.: A genetic algorithm for the weight setting problem in OSPF routing. J. of Combinatorial Optimization **6**, 299–333 (2002)
38. Farago, A., Szentesi, A., Szviatovszki, B.: Allocation of administrative weights in PNNI. In: Proceedings of Networks '98, Sorrento, Italy, pp. 621–625 (1998)
39. Farago, A., Szentesi, A., Szviatovszki, B.: Inverse optimization in high-speed networks. Discrete Applied Mathematics **129**, 83–98 (2003)
40. Fekete, S., Hochstättler, W., Kromberg, S., Moll, C.: The complexity of an inverse shortest path problem. In: R. Graham, J. Kratochvíl, J. Nesetril, F. Roberts (eds.) Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future, vol. 49, pp. 113–127. AMS (1999)

41. Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J.: NetScope: Traffic engi-
    neering for IP networks. IEEE Transactions on Networking **14**, 11–19 (2000)
42. Fortz, B., Thorup, M.: Internet traffic engineering by optimizing OSPF weights. In: Proceed-
    ings of IEEE INFOCOM'00, pp. 519–528 (2000)
43. Fortz, B., Thorup, M.: Optimizing OSPF/IS-IS weights in a changing world. IEEE Journal on
    Selected Areas in Communications **20**(4), 756–767 (2002)
44. Fortz, B., Thorup, M.: Robust optimization of OSPF/IS-IS weights. In: W. Ben-Ameur,
    A. Petrowski (eds.) Proc. INOC 2003, pp. 225–230 (2003)
45. Fortz, B., Thorup, M.: Increasing Internet capacity using local search. Computational Opti-
    mization and Applications **29**(1), 13–48 (2004)
46. Gajowniczek, P., Pióro, M., Szentesi, A., Harmatos, J., Jüttner, A.: Solving an OSPF routing
    problem with simulated allocation. In: Proceedings of 1st Polish-German Teletraffic Sympo-
    sium, pp. 177–184. Dresden, Germany (2000)
47. Gerla, M., Kleinrock, L.: Communication nets: stochastic message flow and delay **25**(1), 48–
    60 (1977)
48. de Giovanni, L., Fortz, B., Labbé, M.: A lower bound for the Internet protocol network design
    problem. In: Proceedings of the 2nd International Network Optimization Conference (INOC
    2005), Lisbon, Portugal, vol. 1, pp. 402–408 (2005)
49. Gourdin, E.: Optimizing internet networks. ORMS Today **28**(2), 46–49 (2001)
50. Gourdin, E., Klopfenstein, O.: Comparison of different QoS-oriented objectives for multi-
    commodity flow routing optimization. In: Proceedings of the International Conference on
    Telecommunications (ICT 2006) (2006)
51. Holmberg, K., Yuan, D.: A Lagrangian heuristic based branch-and-bound approach for the
    capacitated network design problem. Operations Research **48**, 461–481 (2000)
52. Holmberg, K., Yuan, D.: Optimization of Internet protocol network design and routing. Net-
    works **43**(1), 39–53 (2004)
53. ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA:
    CPLEX 11.0 Reference Manual (2007). URL http://www.cplex.com
54. Jüttner A. Szentesi, A., Harmatos, J., Pióro, M.: On solvability of an OSPF routing problem.
    In: Proc. 15th Nordic Teletraffic Seminar (2000)
55. Karas, P., Pióro, M.: Optimization problems related to the assignment of administrative
    weights in the IP networks routing protocols. In: Proceedings of the 1st Polish-German Tele-
    traffic Symposium PGTS'2000, pp. 185–192 (2000)
56. Klopfenstein, O., Mamy, S.: Choosing weights for IP network dimensioning optimization. In:
    Proc. of the International Symposium on Computers and Communications (ISCC 2006), pp.
    994–999 (2006)
57. Leduc, G., Abrahamsson, H., Balon, S., Bessler, S., D'Arienzo, M., Delcourt, O., Domingo-
    Pascual, J., Cerav-Erbas, S., Gojmerac, I., Masip, X., Pescaph, A., Quoitin, B., Romano, S.,
    Salvatori, E., Skivée, F., Tran, H., Uhlig, S., Ümit, H.: An open source traffic engineering
    toolbox. Computer Communications, **29**(5), 593–610 (2006)
58. Lin, F., Wang, J.: Minimax open shortest path first routing algorithms in networks suporting
    the SMDS service. In: Proceedings of the IEEE International Conference on Communications
    1993 (ICC'93), Geneva, Suisse, vol. 2, pp. 666–670 (1993)
59. Menth, M., Hartmann, M., Martin, R.: Robust IP link costs for multilayer resilience. In:
    Networking. Atlanta, GA, USA (2007)
60. Milbrandt, J.: Possibilities of routing optimization in IP networks. Master's thesis, Department
    of Computer Science, University of Würzburg (2001)
61. Moy, J.: OSPF: Anatomy of an Internet Routing Protocol. Addison-Wesley (1998)
62. Mulyana, E., Killat, U.: An alternative genetic algorithm to optimize OSPF weights. In: Inter-
    net Traffic Engineering and Traffic Management, 15th ITC Specialist Seminar, pp. 186–192
    (July 2002). Würzburg, Germany
63. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. John Wiley & Sons
    (1988)

64. Nucci, A., Schroeder, B., Bhattacharyya, S., Taft, N., Diot, C.: IGP link weight assignment for transient link failures. In: Proceedings of 18th International Teletraffic Congress, pp. 321–330 (2003)
65. Oran, D.: OSI IS-IS intra-domain routing protocol. Internet RFC 1142 (1990). Http://www.ietf.org/rfc/rfc1142.txt
66. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0–Survivable Network Design Library. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007). http://sndlib.zib.de
67. Parmar, A., Ahmed, S., Sokol, J.: An integer programming approach to the OSPF weight setting problem. Optimization Online (2005)
68. Pióro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Morgan Kaufman (2004)
69. Pióro, M., Szentesi, A., Harmatos, J., Jüttner, A.: On OSPF related network optimization problems. In: 8th IFIP Workshop on Performance Modelling and Evaluation of ATM & IP Networks, pp. 70/1–70/14. Ilkley, UK (2000)
70. Pióro, M., Szentesi, A., Harmatos, J., Jüttner, A., Gajowniczek, P., Kozdrowski, S.: On OSPF related network optimization problems. Performance Evaluation **48**, 201–223 (2002). (A preliminary version of this paper appeared in the proc. IFIP ATM IP 2000, Ilkley, England, July 2000)
71. Pióro, M., Tomaszewski, A.: Feasibility issues in shortest-path routing with traffic flow split. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007)
72. Pióro, M., Tomaszewski, A., Dzida, M., Mycek, M., Zagożdżon, M.: Literature survey, models, polyhedral results, and exact and heuristic methods for shortest-path routing problems. Technical report, Institute of Telecommunications, Warsaw University of Technology (2007)
73. Prytz, M.: On optimization in design of telecommunications networks with multicast and unicast traffic. Ph.D. thesis, Royal Institute of Technology, Stockholm, Sweden (2002)
74. Ramalingam, G., Reps, T.: An incremental algorithm for a generalization of the shortest-path problem. Jounal of Algorithms **21**(2), 267–305 (1996)
75. Rexford, J.: Handbook of Optimization in Telecommunications, chap. Route optimization in IP networks (2006)
76. Srivastava, S., Agrawal, G., Pióro, M., Medhi, D.: Determining link weight system under various objectives for OSPF networks using a lagrangean relaxation-based approach. IEEE e-Transactions on Network and Service Management **2**(1), 9–18 (2005)
77. Staehle, D., Köhler, S., Kohlhaas, U.: Towards an optimization of the routing parameters for IP networks. Technical Report TR 258, Department of Computer Science, University of Würzburg (2000)
78. Tomaszewski, A., Pióro, M., Dzida, M., Mycek, M., Zagożdżon, M.: Valid inequalities for a shortest-path routing optimization problem. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007)
79. Tomaszewski, A., Pióro, M., Dzida, M., Mycek, M., Zagożdżon, M.: Optimization of administrative weights in IP networks using the branch-and-cut approach. In: Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal, vol. 2, pp. 393–400 (2005)
80. Tong, C., Lam, K.: An embedded connectionist approach for the inverse shortest paths problem. Technical report, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong (1996)
81. Ümit, H., Fortz, B.: Fast heuristic techniques for intra-domain routing metric optimization. In: Proc. INOC 2007 (2007)
82. Wang, Y., Wang, Z., Zhang, L.: Interenet traffic engineering without full mesh overlaying. In: Proc. INFOCOM'2001 (2001). New York, USA
83. Wolsey, L.: Integer Programming. John Wiley & Sons (1998)
84. Xiao, X., Hannan, A., Bailey, B., Ni, L.: Traffic engineering with MPLS in the Internet. IEEE Network **14**, 28–33 (2000)

Chapter 9
Game-Theoretic Approaches to Optimization Problems in Communication Networks

Vittorio Bilò, Ioannis Caragiannis, Angelo Fanelli, Michele Flammini, Christos Kaklamanis, Gianpiero Monaco, and Luca Moscardelli

Abstract In this chapter we consider fundamental optimization problems arising in communication networks. We consider scenarios where there is no central authority that coordinates the network users in order to achieve efficient solutions. Instead, the users act in an uncoordinated and selfish manner and reach solutions to the above problems that are consistent only with their selfishness. In this sense, the users act aiming to optimize their own objectives with no regard to the globally optimum system performance. Such a behavior poses several intriguing questions ranging from the definition of reasonable and practical models for studying it to the quantification of the efficiency loss due to the lack of users' cooperation. We present several results we achieved recently in this research area and propose interesting future research directions.

Key words: non-cooperative networks, strategic games, Nash equilibria, price of anarchy, price of stability

Vittorio Bilò

Department of Mathematics, University of Salento, Provinciale Lecce-Arnesano, P.O. Box 193, 73100 Lecce, Italy, e-mail: `vittorio.bilo@unile.it`

Ioannis Caragiannis · Christos Kaklamanis
Computer Technology Institute & Department of Computer Engineering and Informatics, University of Patras, 26500 Rio, Greece, e-mail: `caragian, kakl@ceid.upatras.gr`

Angelo Fanelli · Michele Flammini · Gianpiero Monaco · Luca Moscardelli
Dipartimento di Informatica, Università degli Studi dell'Aquila, Via Vetoio, Loc. Coppito, 67100 L'Aquila, Italy, e-mail: `angelo.fanelli,flammini,gianpiero.monaco,moscardelli@di.univaq.it`

9.1 Introduction

In the last few years, a mutual interest between the community of computer sci-
entists and of economists is emerging, especially concerning the relationships be-
tween Distributed Systems and Game Theory. The trigger for this synergy is the
evolution of uncoordinated and non-cooperative communication networks such as
the Internet, peer-to-peer networks, and wireless ad hoc networks, and the conse-
quent diversity of the owners of the relative physical components (routers, comput-
ing devices, communication channels, etc.). This has determined a radical change
in the definition of what distributed computing means. Indeed, in the classical com-
putational model adopted in Theoretical Computer Science, processors are faithful
executors of an algorithm. Conversely, in the incentive-based model, which is the
one traditionally studied by economists, processors (usually called agents) pursue
a selfish strategy and the system evolves as a consequence of the complex interac-
tions among them. Therefore, Game Theory comes to pursue the synthesis between
two requirements: the system designer's or operator's (whose goal is to compute a
socially efficient solution), and the agents' (which, in a non-cooperative scenario,
aim to maximize their own profit and, therefore, could induce the system to reach
suboptimal solutions).

Let us introduce informally the concepts studied in this chapter with an example.
Assume that we have a simple communication network in which two nodes s and t
are connected through two directed links e_1 and e_2 (see Figure 9.1). There are two
users, each having a unit amount of traffic. Each user wishes to route her traffic from
s to t unsplit, using one of the two links. Link e_1 has a latency function $d_{e_1}(x) = x$,
which means that the latency experienced by each user that uses this link is x when
x users in total use this link. The latency function of link e_2 is $d_{e_2}(x) = 2 + \varepsilon$, where
ε is a negligible but strictly positive constant. A natural objective is to minimize
the average (or total) latency among the two users. So, from the network designer
point of view, a solution which forces one user to use link e_1 and the other to use
link e_2 would be the desired one with a total latency of $3 + \varepsilon$. However, if each
user is selfish in the sense that she prefers to use the link which minimizes her
latency given the decision of the other user, we could never end up with this desired
solution. The user that is assigned to link e_2 (where she experiences a latency of
$2 + \varepsilon$) has an incentive to change her strategy and use link e_1 instead; then, the
latency experienced would be only 2. From this latter assignment, no user has an
incentive to unilaterally deviate; such a solution is consistent with the selfish nature
of the behavior of the users. Unfortunately, selfishness may result in deterioration
of system performance; the total latency in this latter solution is 4 since both users
experience a latency of 2 on link e_1.

In the above example, we have implicitly assumed that users (called players in
the following) play a strategic game which reflects their selfish behavior. Solutions
that are consistent with this behavior are called equilibria; these notions are for-
mally defined in the next section. Issues related to the existence, the structure, the
computation, and the efficiency of equilibria in strategic games arising from opti-
mization problems are studied (among other issues) in the recently emerging field
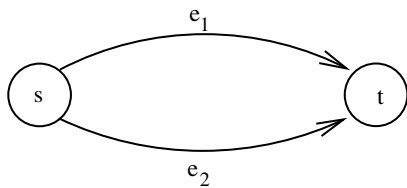
Fig. 9.1 A simple communication network

of Algorithmic Game Theory. In this chapter we present some basic contributions in this fascinating emerging area related to some fundamental optimization problems that arise in communication networks with non-cooperative users. We focus on results which we have obtained recently; the interested reader may see [39] for a more systematic study of the recent trends in Algorithmic Game Theory.

9.2 Preliminary Notions

We begin with some necessary definitions. Given a set U, a k-tuple $a = (u_1, \ldots, u_k)$ of elements of U, an index $i \in \{1, \ldots, k\}$, and an element $u \in U$, we write $(a_{-i}, u) = (u_1, \ldots, u_{i-1}, u, u_{i+1}, \ldots, u_k)$ to denote the k-tuple obtained from a by replacing u_i with u.

Strategic games. A strategic game $\mathscr{G}$ is a triple $\mathscr{G} = (P, S_{i \in P}, \omega_{i \in P})$ where P is a set of n players, S_i is the set of strategies available to player i and $\omega_i : S_1 \times \ldots \times S_n \mapsto \mathbf{R}$ is her payoff function. The payoff function ω_i can model either a benefit or a cost for player i; thus each player may seek the maximization or the minimization of her payoff. In the sequel we will always assume that the payoff functions model costs for the players.

States and improving steps. The set $S = S_1 \times \ldots \times S_n$ is called the set of states or strategy profiles of $\mathscr{G}$. Consider a state $\sigma = (\sigma_1, \ldots, \sigma_n) \in S$. A player i cannot be happy with her payoff in σ if there exists another state $\sigma' = (\sigma_{-i}, s)$, for some $s \in S_i$, such that $\omega_i(\sigma') < \omega_i(\sigma)$. The action of changing their strategy from σ_i to s (with the consequent transition of $\mathscr{G}$ from state σ to state σ') is called an improving step or a selfish move performed by player i.

Pure Nash equilibria. A very important and challenging issue in game theory is to characterize solutions of games which are consistent with the selfish and rational behavior of the players. Among all the approaches proposed so far, the notion of Nash equilibrium is the most accepted and used one. A state σ is a pure Nash equilibrium if no player has an improving step, that is, $\forall i \in P$ and $\forall s \in S_i$ it holds $\omega_i(\sigma_{-i}, s) \geq \omega_i(\sigma)$. A pure Nash equilibrium is a stable outcome of a game in the sense that it is a state in which all players are satisfied with their payoff and none

of them can improve it by unilaterally changing her strategy. The drawback of this notion is, however, due to the fact that pure Nash equilibria are not guaranteed to exist for any game.

Mixed strategies. A mixed strategy for player i is a probability distribution Y_i defined over the set S_i of her pure strategies. Given a mixed strategy Y_i for player i, the support of Y_i, denoted as $support(Y_i)$, is the set of strategies $s \in S_i$ for which $Y_i(s) > 0$. A mixed strategy $Y = (Y_1 \ldots, Y_n)$ is an n-tuple of mixed strategies Y_i for each player $i \in P$. The support of a mixed strategy Y is defined as the set of states $support(Y) = support(Y_1) \times \ldots \times support(Y_n)$. The payoff of player i yielded by a mixed strategy Y is defined as $\omega_i(Y) = \sum_{\sigma \in support(Y)}(\omega_i(\sigma) \cdot Prob_Y(\sigma))$, where $Prob_Y(\sigma) = \prod_{i=1}^{n} Y_i(\sigma_i)$.

Mixed Nash equilibria. A mixed strategy Y is a (mixed) Nash equilibrium if $\forall i \in P$ and for any probability distribution Z defined over S_i it holds $\omega_i(Y_{-i}, Z) \geq \omega_i(Y)$. A fully mixed Nash equilibrium Y is a mixed Nash equilibrium such that $support(Y_i) = S_i$ for each $i \in P$. Mixed Nash equilibria are, hence, a generalization of pure ones, obtained by allowing players to randomize on their chosen strategies and computing the resulting payoffs in expectation. In this case the solution concept becomes less reasonable (since, in practice, we are always asked to choose pure strategies); but, on the other hand, Nash's famous theorem [38] guarantees the existence of at least one mixed Nash equilibrium for any game.

Price of anarchy and stability. One of the main concerns when dealing with non-cooperative systems is to bound their inefficiencies due to the lack of coordination among the players. More formally, given a function $\gamma : S \mapsto \mathbf{R}$, called the social function, let σ^* be a state optimizing γ. On the other hand, given a mixed strategy Y, the social value of Y is defined as $\gamma(Y) = \sum_{\sigma \in support(Y)}(\gamma(\sigma) \cdot Prob_Y(\sigma))$. The price of anarchy [35] of game $\mathscr{G}$ for the social function γ is defined as $PoA(\mathscr{G}, \gamma) = \sup_{Y \in NE(\mathscr{G})} \frac{\gamma(Y)}{\gamma(\sigma^*)}$, while the price of stability [3] of $\mathscr{G}$ for the social function γ is defined as $PoS(\mathscr{G}, \gamma) = \inf_{Y \in NE(\mathscr{G})} \frac{\gamma(Y)}{\gamma(\sigma^*)}$, where $NE(\mathscr{G})$ denotes the set of Nash equilibria of $\mathscr{G}$. By restricting $NE(\mathscr{G})$ to the set of pure Nash equilibria, similar definitions for the price of anarchy and stability are obtained for pure strategies. The price of anarchy is a classical worst-case analysis and it measures the loss of performance due to the selfish behavior of players in this case. On the other hand, the price of stability gives us information on the minimum loss of performance a non-cooperative system has to suffer. This issue is best understood when dealing with games, such as network design games, in which it is possible to assume the presence of a central authority proposing or imposing a pure Nash equilibrium to the players. In this setting, the best Nash equilibrium represents the optimal solution among all the proposed ones from which no player would have an incentive to defect. By extending the above definition, the price of anarchy after a best response walk can be defined as the worst-case ratio of the social value of the last state of the walk to the value of the social optimum.

Nash dynamics graph. For any game $\mathcal{G}$ it is possible to define a graph $D(\mathcal{G}) = (S, A)$ capturing the dynamic behavior of the players in the game. The set of nodes of $D(\mathcal{G})$ is represented by the set of states of $\mathcal{G}$, and there is a directed edge between any two nodes σ and σ' with label i if and only if player i has an improving step from σ to σ' in $\mathcal{G}$. By definition, we have that $\mathcal{G}$ possesses pure Nash equilibria if and only if $D(\mathcal{G})$ has sink nodes, that is, nodes without any outgoing edge. An important issue related to the notion of pure Nash equilibria is the finite improvement path (FIP) property. A game has this property if, starting from any initial state and letting a player perform arbitrary improving steps, a pure Nash equilibrium is always reached, i.e., there does not exist an infinite sequence of improving steps. Again, by definition, we have that $\mathcal{G}$ has the FIP property if and only if $D(\mathcal{G})$ is acyclic, that is, a DAG (directed acyclic graph). In a given state a player i may have more than one improving step, i.e., for a given node there may exist more than one outgoing edge labeled i in $D(\mathcal{G})$. An improving step for player i in state σ is a best response for i in σ if it yields the maximum improvement in i's payoff among all improving steps i has in σ. We define the best response dynamics graph of $\mathcal{G}$ as the graph obtained from $D(\mathcal{G})$ by removing all edges not modeling best responses.

Best response walks. Any path in the best response dynamics graph is called a *best response walk*. As there are games that do not have the FIP property as well as games for which the number of best responses (or, more generally, improving steps) needed to reach an equilibrium starting from an arbitrary state may be exponentially large, it becomes important to evaluate the social value of states reached after a finite number of selfish moves. To this aim, Mirrokni and Vetta [36] introduced the following models capturing the intuitive notion of a fair sequence of moves.

> *Covering walk.* A walk in the best response dynamics graph is a covering walk if for each player i, it has at least one edge with label i.
> *k-Covering walk.* A walk in the best response dynamics graph is a *k*-covering walk if it can be split into k disjoint covering walks.
> *One-round walk.* A covering walk of length n in the best response dynamics graph is a one-round walk, that is, each player performs exactly one best response.

Potential games. If $\mathcal{G}$ has the FIP property then, by exploiting the topological ordering of the nodes in $D(\mathcal{G})$, it is possible to define a potential function for $\mathcal{G}$, that is, a function $\Phi : S \mapsto \mathbf{R}$ always decreasing or always increasing each time an improving step is performed. By generalizing this observation, it is possible to define the class of potential games as the class of all games having a potential function and, hence, the FIP property. Depending on the properties satisfied by Φ, three different kinds of potential games can be defined.

1. *Exact potential* games, where $\forall \sigma \in S$ and $\forall s \in S_i$, it holds $\Phi(\sigma) - \Phi(\sigma_{-i}, s) = \omega_i(\sigma) - \omega_i(\sigma_{-i}, s)$.

2. *Weighted potential* games, where $\exists \beta = (\beta_1, \ldots, \beta_n)$ such that $\forall \sigma \in S$ and $\forall s \in S_i$, it holds $\Phi(\sigma) - \Phi(\sigma_{-i}, s) = \beta_i(\omega_i(\sigma) - \omega_i(\sigma_{-i}, s))$.

3. *Ordinal potential* games, where $\forall \sigma \in S$ and $\forall s \in S_i$, it holds that $\Phi(\sigma) - \Phi(\sigma_{-i}, s)$ and $\omega(\sigma) - \omega(\sigma_{-i}, s)$ have the same sign.

Congestion games. Perhaps the most famous class of games is that of congestion games. Results concerning congestion games are presented in Sections 9.3 and 9.7. Here, we present the basic related definitions. A congestion game is a 4-tuple $(P, E, S_{i \in P}, d_{e \in E})$, where P is a set of n players, E is a set of m resources, $S_i \subseteq 2^{|E|}$ for each $i \in P$, and $d_e : \mathbf{N} \mapsto \mathbf{R}$ for each $e \in E$. Strictly speaking, each player in a congestion game can choose among different subsets of resources; each resource e has an associated delay (latency) function d_e returning the delay experienced by any player using e in terms of the number of players using it. Once $n_e(\sigma) = |\{i \in P : e \in \sigma_i\}|$ is defined as the number of players using resource e in state σ, the payoff function of player i is defined as $\omega_i(\sigma) = \sum_{e \in \sigma_i} d_e(n_e(\sigma))$. Congestion games were introduced by Rosenthal [41], who proved that they have the FIP property by defining the potential function $\Phi(\sigma) = \sum_{e \in E} \sum_{i=1}^{n_e(\sigma)} d_e(i)$. As shown by Monderer and Shapley [37], the class of congestion games is equivalent to that of exact potential games. The class of weighted congestion games is a generalization obtained by allowing players to have different demands. In this case the congestion of each resource e is defined as $n_e(\sigma) = \sum_{i \in P : e \in \sigma_i} r_i$, where r_i is the demand of player i. Some special cases of congestion games are particularly interesting to computer scientists. In network congestion games, resources correspond to the links of a network. Each player i has a source s_i and the destination t_i and her set of strategies corresponds to the set of paths connecting s_i to t_i. Load balancing games are congestion games in which all possible strategies for each player are singletons.

Cost sharing games. Similarly to congestion games, a cost sharing game is a 4-tuple $(P, E, S_{i \in P}, c_{e \in E})$, where P is a set of n players, E is a set of m resources, $S_i \subseteq 2^{|E|}$ for each $i \in P$, and $c : E \mapsto \mathbf{R}^+$. In these games, each resource e has a certain fixed cost c_e, which has to be shared among the players. In particular, each strategy profile σ requires the use of a set of resources $\Pi(\sigma) = \bigcup_{i=1}^{n}\{e \in E : e \in \sigma_i\}$ of total cost $cost(\Pi(\sigma)) = \sum_{e \in \Pi(\sigma)} c_e$, and the payoffs yielded by σ are computed by applying a certain cost sharing method $\mathcal{M}$ distributing $cost(\Pi(\sigma))$ among the players. Depending on the definition of $\mathcal{M}$, different games can be obtained. The most important family of cost sharing games arises in network design where the cost of a certain self-emerging network needs to be shared among its users. In this case, there is an edge-weighted graph $G = (V, E, c)$, with $c : E \mapsto \mathbf{R}^+$, modeling the set of links which can be potentially implemented, as well as their building costs. Each player i has a source s_i and a destination t_i that she wants to connect, and her set of strategies corresponds to the set of paths connecting s_i to t_i in G. This general network cost sharing game is referred to as the multi-commodity case. An interesting and much studied restriction is the single-commodity case, also called the multicast case, in which there is a common source s for all players and a set

$R \subset V \setminus \{s\}$ of n receivers which need to be connected to s. The most-used cost sharing method is the one based on the well-known Shapley value formula [45] which equally distributes the cost of each resource among the players using it, thus yielding $\omega_i(\sigma) = \sum_{e \in \sigma_i} \frac{c_e}{n_e(\sigma)}$. A cost sharing game in which the underlying cost sharing method is the Shapley value is called a Shapley cost sharing game. As first noted in [2] these games are also congestion games. Results related to cost sharing games are covered in Sections 9.4 and 9.7.

9.3 Congestion Games

In this section we consider congestion games. Besides general congestion games, we also focus on network congestion games and load balancing games.

   We study issues related to the efficiency of equilibria in congestion games according to the social cost function of the weighted total latency, i.e., the sum of the latency experienced by each player multiplied by her demand. Formally, the social cost of a state σ is $\gamma(\sigma) = \sum_i r_i \omega_i(\sigma)$. An equivalent definition is $\gamma(\sigma) = \sum_e n_e(\sigma) d_e(n_e(\sigma))$, since the sum of the demands of the players using resource e is $n_e(\sigma)$ and each of them experiences a latency of $d_e(n_e(\sigma))$ on e.

Price of anarchy. A large part of the literature related to congestion games is devoted to providing upper and lower bounds on the price of anarchy. What we essentially want to do in this case is to relate the maximum social cost over all equilibria to the optimal social cost. The following simple argument is the heart of most related proofs in the literature. Consider an equilibrium σ for a congestion game and let σ^* be the assignment of optimal social cost. Assume that the latency functions are linear with the simple form $d_e(x) = \alpha_e x$ and that the demands of the players are the same (and equal to 1). Since no player has an incentive to change her strategy in σ, this also means that $\omega_i(\sigma) \leq \omega_i(\sigma_{-i}, \sigma_i^*)$, i.e., that the player i has no incentive to change her strategy to the one it uses in σ^*. Substituting the payoff, this yields that

$$\sum_{e \in \sigma_i} d_e(n_e(\sigma)) \leq \sum_{e \in \sigma_i^*} d_e(n_e(\sigma_{-i}, \sigma_i^*)) \leq \sum_{e \in \sigma_i^*} d_e(n_e(\sigma) + 1).$$

The second inequality follows since $(\sigma_{-i}, \sigma_i^*)$ and σ differ only in the strategy of player i and since the latency functions are non-decreasing. Then, the social cost of σ is

$$\gamma(\sigma) = \sum_i \omega_i(\sigma)$$

$$= \sum_i \sum_{e \in \sigma_i} d_e(n_e(\sigma))$$

$$\leq \sum_i \sum_{e \in \sigma_i^*} d_e(n_e(\sigma)+1)$$

$$= \sum_e \sum_{i:e \in \sigma_i^*} d_e(n_e(\sigma)+1)$$

$$= \sum_e n_e(\sigma^*) d_e(n_e(\sigma)+1)$$

$$= \sum_e \alpha_e \left(n_e(\sigma^*) n_e(\sigma) + n_e(\sigma^*) \right) \tag{9.1}$$

The next step is to apply a simple inequality such as $xy + y \leq \frac{1}{3}x^2 + \frac{5}{3}y^2$ for any integer $x, y \geq 0$ on the right part of (9.1) to obtain (by setting $x = n_e(\sigma)$ and $y = n_e(\sigma^*)$)

$$\gamma(\sigma) \leq \sum_e \alpha_e \left(\frac{1}{3} n_e^2(\sigma) + \frac{5}{3} n_e^2(\sigma^*) \right)$$

$$= \frac{1}{3}\gamma(\sigma) + \frac{5}{3}\gamma(\sigma^*)$$

and, hence, $\gamma(\sigma) \leq \frac{5}{2}\gamma(\sigma^*)$, which means that the price of anarchy is at most $5/2$.

The above bound was independently obtained by Christodoulou and Koutsoupias [21] and Awerbuch et al. [4]. The proof can be generalized to prove that the price of anarchy of weighted linear congestion games over mixed Nash equilibria is at most $\phi^2 \approx 2.62$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. These results are tight. Several papers in the literature (e.g., [1, 4, 21, 46]) have applied the same technique for more general (e.g., polynomial) latency functions.

Prior to these works, Suri et al. [46] had obtained the same bound for load balancing games. A natural question in load balancing games is whether their simplicity (compared to congestion games) leads to better bounds on the price of anarchy. In [15], it has been shown that this is not the case in general by proving tight lower bounds of 2.62 and $5/2$ on the price of anarchy of load balancing games with linear latency functions and players with different or equal demands, respectively. However, there are cases where load balancing games have some special structure that decreases the price of anarchy. For example, [15] exploits information about the structural properties of load balancing games on machines with identical linear latency functions to show an upper bound of 2.012 for the price of anarchy over pure Nash equilibria. This result is tight; a matching lower bound had been previously obtained in [46].

Price of stability. The argument that is used to prove upper bounds on the price of stability is similar. Besides the use of the Nash inequality, information provided by the potential function is also used. Essentially, we aim to relate the social cost of the best pure Nash equilibrium σ' to the social cost of the optimal assignment σ^*. We

know that the social cost of σ' is upper-bounded by the social cost of an equilibrium σ which can be reached in the Nash dynamics graph by following improving steps from σ^*. Hence, in order to compute an upper bound on the price of stability, it suffices to compute an upper bound on the social cost of σ in terms of the optimal social cost of σ^*. By the definition of the potential function, we know that the potential of σ is smaller than the potential of σ^*, i.e., $\Phi_R(\sigma) \le \Phi_R(\sigma^*)$. Again, let us assume that the latency functions have the simple linear form $d_e(x) = \alpha_e x$ and that the demands of the players are the same (and equal to 1). Then, the potential of σ can be written as

$$\Phi_R(\sigma) = \sum_e \sum_{j=1}^{n_e(\sigma)} d_e(j) = \frac{1}{2} \sum_e \alpha_e \left(n_e^2(\sigma) + n_e(\sigma) \right)$$

and the inequality of potentials can be used to obtain the following derivation.

$$\begin{aligned} \gamma(\sigma) &= \sum_e \alpha_e n_e^2(\sigma) \\ &\le \sum_e \alpha_e \left(n_e^2(\sigma) + n_e(\sigma) - n_e(\sigma^*) \right) \end{aligned} \tag{9.2}$$

The main idea in the proofs of [22] and [15] is to use the information provided by both (9.1) and (9.2) in order to bound the social cost of σ in terms of the optimal social cost. This requires multiplying each of the two inequalities with a particular coefficient, summing them, and then using an inequality on integers in order to bound the right part by an expression that contains only the social cost of σ and the optimal social cost of σ^*. In this way, an upper bound of $1 + \frac{1}{\sqrt{3}} \approx 1.577$ on the price of stability is proved in [15] (prior to this result, a slightly inferior bound of 1.6 had been presented in [22]). This bound is tight; a congestion game with a single equilibrium of social cost $1 + \frac{1}{\sqrt{3}}$ times the optimal social cost has been presented in [22].

In load balancing games with linear latency functions, the price of stability has been proved to be much smaller, namely $4/3$. The lower bound easily follows by a simple game with two players of unit demands on two resources with latency functions $f_{e_1}(x) = x$ and $f_{e_2}(x) = (2+\varepsilon)x$ for arbitrarily small $\varepsilon > 0$. The state in which both players select resource e_1 is the only pure Nash equilibrium of social cost 4 while the states in which the players select different resources have social cost $3 + \varepsilon$. In order to prove the upper bound, the information provided by the potential function is not enough. The approach followed in [15] is to provide a particular sequence of improving steps from a state σ^* of optimal social cost to a pure Nash equilibrium σ. The sequence is defined in such a way that a comparison between the social costs of σ and σ^* is feasible. The interested reader can inspect [15] for the details of the technique.

Independently, Anshelevich et al. [2] have shown a more general result that characterizes the price of stability of any network congestion game in which players have a common source (or a common destination). Namely, the result of [2] states that the price of stability of these games is at most the price of anarchy of corresponding

nonatomic congestion games [44] on the same network. The main assumption in these games which differentiates them from the games we study in this section is that the number of players is infinite, and each is supposed to control a negligible amount of traffic from its source to its destination.

Performance of k-round walks. Convergence issues have been the subject of several papers. In [24], the authors show an upper bound of $\Theta(n)$ on the price of anarchy after one round, and a lower bound of $2^{\mathcal{O}(k)}\sqrt{n}/k$ after k rounds. Convergence to a constant price of anarchy in a polynomial number of random rounds can be inferred directly from the sink equilibria results of [36], while the $2^{\mathcal{O}(k)}\sqrt{n}/k$ lower bound of [24] implies that a number of rounds at least proportional to $\log\log n$ is necessary. In [27] it is shown that the price of anarchy achieved after k rounds is $\mathcal{O}\left(\sqrt[2^{k-1}]{n}\right)$ while the lower bound of [24] is refined to $\Omega\left(\sqrt[2^{k-1}]{n}/k\right)$, which is asymptotically matching for constant values of k. As a consequence, $\log\log n$ rounds are not only necessary, but also sufficient in order to achieve a constant price of anarchy, i.e., comparable to the one at Nash equilibrium. [27] also provides a new lower bound of $\Omega\left(\sqrt[2^k-1]{n}\right)$ for load balancing games, thus showing that a number of rounds proportional to $\log\log n$ is necessary and sufficient under such a restriction.

Coping with selfishness. Another line of research in congestion games aims to cope with selfishness. There are at least four different approaches that have been proposed in the literature: coordination mechanisms [6, 14, 23, 31], Stackelberg strategies [42], network design or resource removal [5, 43], and taxes or tolls [16, 17, 25, 29, 32]. The general idea behind each of them is to change the congestion game at hand in a reasonable way so that the resulting game has a small price of anarchy. In network design [43], some of the resources are removed; this also constrains the sets of strategies of the players. The results of Azar and Epstein [5] indicate that even deciding whether network design can decrease the price of anarchy of weighted network congestion games with linear latency functions to significantly less than ϕ^2 (the upper bound on the price of anarchy of such games) is NP-hard. Besides this, they show that there exist games in which network design is not helpful at all.

In the following, we briefly discuss the results in [16], which seems to be the first work dealing with taxes in atomic congestion games. In contrast, the study of taxes in nonatomic congestion games has a long history, starting at the beginning of the twentieth century with the work of Pigou [40]; and, besides recent contributions from Computer Science [25, 29, 32], it contains contributions from Economics and Transportation Science.

The model used in [16] is the following. A tax function $\delta : E \times Q^+ \to Q^+$ is introduced, meaning that a tax $\delta_e(w)$ is assigned to each player of demand w wishing to use the resource e. In this way, an extended game is obtained in which the cost of each player is the sum of the latency experienced and the tax she pays. The latency functions considered in [16] are linear. By defining a potential function (similar to the potential function for weighted linear congestion games [30]), it is shown that the extended game always has a pure Nash equilibrium. We note that network design can be thought of as a special kind of tax (infinite tax on some of the resources).

In the case of symmetric load balancing games, pure optimal taxes (i.e., taxes such that any pure Nash equilibrium of the extended game has optimal total latency) exist and can be constructed in polynomial time. Besides this positive result, even in simple congestion games, pure optimal taxes do not exist. Consider the following simple congestion game with four resources e_1, e_2, e_3, e_4 with the same latency function $d(x) = x$ and four players of demand 1: two long players, each having strategies $\{e_1, e_3\}, \{e_2, e_4\}$ and two short players, each having strategies $\{e_1\}, \{e_2\}$. Observe that assignments in which each of the resources e_1 and e_2 is used by one long and one short player have optimal total latency 10. Consider a tax assignment δ. It can be easily verified that if $\delta_{e_1} + \delta_{e_3} \leq \delta_{e_2} + \delta_{e_4}$, the assignment where the long players select strategy $\{e_1, e_3\}$ and the short players select strategy $\{e_2\}$ is a pure Nash equilibrium for the extended game; otherwise, the assignment where the long players select strategy $\{e_2, e_4\}$ and the short players select strategy $\{e_1\}$ is a pure Nash equilibrium for the extended game. In any case, the total latency is 12. Even simpler load balancing games do not admit pure optimal taxes; the reader can inspect [16] and [17] for more such negative statements.

Since taxes cannot always force optimal system performance, the next step is to study whether there are taxes such that the extended game has efficient equilibria. The efficiency of these equilibria (mixed or pure) is assessed in two different ways. In the case of refundable taxes, the social cost is simply the (weighted) total latency of the players. In the case of nonrefundable taxes, the social cost is the (weighted) total latency plus the total taxes paid. Assumptions similar to those in refundable taxes are common in the study of taxes for nonatomic congestion games in the Economics and Transportation literature and capture the scenarios where the collected taxes can be feasibly returned (directly or indirectly) to the players (e.g., as a "lump-sum refund").

The terms ρ-pure-efficient and ρ-mixed-efficient are used to refer to taxes which guarantee that the social cost of each equilibrium of the extended game is at most ρ times larger than the optimal (weighted) total latency. Again, a negative result shows that very simple symmetric load balancing games on identical machines do not admit better than 2-mixed-efficient taxes. On the positive side, 2-mixed-efficient refundable taxes with respect to the (weighted) total latency can be computed in polynomial time using convex quadratic programming. The case of nonrefundable taxes is more difficult to handle. However, there are load balancing games where ρ-mixed-efficient nonrefundable taxes can be computed for values of ρ which are smaller than the price of anarchy of the original game (without taxes). The interested reader can refer to [16] and [17] for the related results, proofs, and open problems.

9.4 Multicast Cost Sharing Games

In this section, we consider multicast cost sharing games. A *cost sharing method* for multicast games is a function $\mathcal{M}$ which, given a set of receivers R and a strategy profile σ, distributes among the receivers the total cost $cost(\Pi(\sigma))$ in such a

way that $\sum_{i \in R} \mathcal{M}(R, \sigma, i) = cost(\Pi(\sigma))$, where $\mathcal{M}(R, \sigma, i) =_{def} \omega_i(\sigma)$ is the cost charged to receiver i.

Several cost sharing methods have been proposed so far, namely,

- $\mathcal{M}_1$ (*egalitarian*) equally shares the global cost among all the receivers, i.e.,

$$\mathcal{M}_1(R, \sigma, i) = \frac{cost(\Pi(\sigma))}{n}.$$

- $\mathcal{M}_2$ (*path-proportional*) shares the cost of each link $e \in \Pi(\sigma)$ among all the receivers j using it proportionally to the overall cost of their chosen path, i.e.,

$$\mathcal{M}_2(R, \sigma, i) = \sum_{e \in \Pi(\sigma_i)} c_e \frac{cost(\Pi(\sigma_i))}{\sum_{j : e \in \Pi(\sigma_j)} cost(\Pi(\sigma_j))}.$$

- $\mathcal{M}_3$ (*egalitarian-path-proportional*) shares the overall cost among all the receivers proportionally to the cost of their chosen path, i.e.,

$$\mathcal{M}_3(R, \sigma, i) = cost(\Pi(\sigma)) \frac{cost(\Pi(\sigma_i))}{\sum_{j \in R} cost(\Pi(\sigma_j))}.$$

- $\mathcal{M}_4$ (*Shapley*) equally shares the cost of each link $e \in \Pi(\sigma)$ among all the receivers using it, i.e.,

$$\mathcal{M}_4(R, \sigma, i) = \sum_{e \in \Pi(\sigma_i)} \frac{c_e}{n_e(\sigma)}.$$

Since cost sharing games naturally arise in socioeconomic scenarios, cost sharing methods are usually required to meet some constraining properties. The ones most standard and used are:

- *Weak budget balance.* A receiver is never charged a cost share greater than the cost of her chosen path, that is, $\mathcal{M}(R, \sigma, i) \leq cost(\Pi(\sigma_i))$.
- *Strong budget balance.* The cost of each link is only shared among the receivers using it.
- *Stability.* The game induced by $\mathcal{M}$ possesses pure Nash equilibria.
- *Fairness.* The cost share charged to any two receivers adopting two equivalent paths in a given strategy profile is the same, where two paths are equivalent if they have the same cost and are shared in the same way by paths having the same cost. More formally, let $a_k(e, \sigma)$ be the set of paths of cost k using edge e in σ; two paths p and p' are equivalent in σ if $cost(p) = cost(p')$ and $\sum_{e \in p \cap a_k(e, \sigma)} c_e = \sum_{e \in p' \cap a_k(e, \sigma)} c_e$ for any $k \geq 0$.
- *Separability.* The cost shares of each edge are computed independently and are completely determined by the set of receivers using it.

We call feasible any method meeting weak budget balance and fairness. Clearly, the strong budget balance property implies the weak one. In [20] it is remarked that the only method meeting all such properties is the Shapley value. Moreover, it is

not difficult to see that the path proportional one meets strong budget balance and fairness, while the egalitarian-path-proportional one meets weak budget balance and fairness. Hence, all the above methods besides the egalitarian one are feasible.

Directed graphs. In [8] it is shown that all methods except for the path-proportional one meet stability. The price of anarchy for the game yielded by the egalitarian method is unbounded, while for all the other ones it is equal to n with respect to two different social cost functions: the overall transmission cost (function γ_{sum}), which coincides with the sum of all the shared costs, and the maximum shared cost paid by the receivers (function γ_{max}). As for the price of stability, in [2] it is proved that it is $\Theta(\log n)$ for the Shapley value with respect to γ_{sum}.

For any feasible method, the number of best response moves needed to reach a Nash equilibrium starting from any strategy profile can be arbitrarily large, even when $n = 2$. Motivated by these results, it becomes interesting to evaluate the price of anarchy after a limited number of best responses. For the egalitarian and path-proportional methods the price of anarchy is unbounded for any sequence of best responses and one-round walks. For the Shapley value method, the price of anarchy after a one-round walk is $\Theta(n^2)$. Such a value is outperformed by the egalitarian-path-proportional method, which achieves a price of anarchy of $\mathcal{O}(n)$ after a one-round walk. This is an asymptotically optimal result since it can be shown that any feasible method cannot achieve a price of anarchy better than n after a one-round walk. All the above results have been determined for both γ_{sum} and γ_{max} in [8].

For k-round walks, [26] shows that, starting from any strategy profile, the Shapley method achieves a price of anarchy of at most $\mathcal{O}(n\sqrt{n})$ after two rounds and gives a general lower bound of $\Omega(n\sqrt[k]{n})$ for any number $k \geq 2$ of rounds. Similarly, when starting from the empty strategy profile, exactly matching upper and lower bounds equal to n are determined for any number of rounds. When starting from an arbitrary strategy profile, both the egalitarian and the path-proportional methods can yield an unbounded price of anarchy, while the egalitarian path-proportional achieves a price of anarchy of $\Theta(n)$. Finally, when starting from the empty strategy profile, all these three methods achieve a price of anarchy of $\Theta(n)$.

Undirected graphs. We briefly describe results related to undirected graphs by outlining the differences with the directed case. When not explicitly claimed, all the presented results are taken from [8]. Deciding whether the path-proportional method meets stability is still an open problem. The price of anarchy of the egalitarian path-proportional method falls between $\frac{2}{3}n$ and n. Asymptotic lower bounds equal to 1.915, 2, and 1.714 are known, respectively, for the price of stability of the path-proportional, the egalitarian path-proportional, and the Shapley value methods with respect to γ_{sum} [7]. The price of anarchy of the egalitarian path-proportional method after a one-round walk at least $\frac{2}{3}n$. No general results on the performance of feasible methods are known. The main differences with the case of directed graphs hold for the best-response walks. For the price of anarchy of the Shapley value, only the trivial lower bound of $\Omega(n)$ is known for k-round walks when starting from any arbitrary strategy profile. When starting from the empty strategy profile, in [18] an

upper bound of $\mathcal{O}(\log^3 n)$ and a lower bound of $\Omega(\log n)$ on the price of anarchy achievable after any number of rounds is proved. For one-round walks, an improved $\Omega(\log^2 n)$ lower bound is derived. For the egalitarian method we have a price of anarchy of $\Theta(\log n)$, for the path-proportional one we have a lower bound of $\Omega(\log n)$ and an upper bound of $\mathcal{O}(n)$, and for the egalitarian path-proportional a lower bound of $\Omega(n/\log n)$ and an upper bound of $\mathcal{O}(n)$.

Finally, finding a Nash equilibrium minimizing the potential function is NP-hard [19], as is finding the sequence of the best responses leading to the lowest possible social cost even after a one-round walk for both γ_{sum} and γ_{max}.

9.5 Communication Games in All-Optical Networks

All-optical networks have been largely investigated in recent years due to the promise of data transmission rates several orders of magnitude higher than those of current networks. The key to high speeds in all-optical networks is to maintain the signal in optical form, thereby avoiding the prohibitive overhead of conversion to and from electrical form at the intermediate nodes. The high bandwidth of the optical fiber is utilized through *wavelength-division multiplexing*: two signals connecting different source-destination pairs may share a link, provided they are transmitted on carriers having different wavelengths (or colors) of light. Since the optical spectrum is a scarce resource, a given communication pattern in optical networks is often designed so as to minimize the total number of colors used, a measure which is trivially lower-bounded by the maximum load, that is, the maximum number of connecting paths sharing the same physical edge.

In [12], the following problem is investigated. An all-optical network provider must determine suitable payment functions for non-cooperative agents wishing to communicate so as to induce Nash equilibria using a low number of wavelengths. More formally, an all-optical network is modeled as an undirected graph $G = (V, E)$ where nodes in V represent sites and undirected edges in E represent bidirectional optical fiber links between the sites. A point-to-point communication requires us to establish a uniquely colored path between the two nodes whose color is different from the colors of all the other paths sharing some of its edges. Each player in the game asks for the implementation of a certain point-to-point communication and is charged a cost by the network provider obtained by applying a certain payment function. Depending on the variables defining the payment function, it is possible to distinguish among three different levels of information needed by an agent in order to compute her cost charge.

- *Minimal*. Each agent i knows all the wavelengths available along any path in S_i.
- *Intermediate*. Each agent i knows the wavelengths available along any edge in the network.
- *Complete*. Each agent i knows the whole strategy profile.

Under the complete level, suitable payment functions can be computed such that in any Nash equilibrium the assignment of paths and colors to the agents is the same as with the one computed by a centralized algorithm aiming to minimize the optical spectrum, and the computational complexity is the same as that of the algorithm. For the remaining two levels, the most reasonable payment functions either do not admit pure Nash equilibria or induce games having the worst possible price of anarchy, that is, always possessing a pure Nash equilibrium assigning a different color to each agent. However, by suitably restricting the network topology, a price of anarchy of 8 has been obtained for chains and 16 for rings under the minimal level, and further reduced respectively to 3 and 6 under the intermediate level, up to an additive factor converging to 0 as the load increases. Finally, again under the minimal level, a price of anarchy logarithmic in the number of agents has been determined for trees.

9.6 Beyond Nash Equilibria: An Alternative Solution Concept for Non-cooperative Games

As witnessed by the related notions of price of anarchy and stability, pure Nash equilibria usually generates suboptimal solutions for non-cooperative games. One of the several reasons for this situation is the fact that players always perform selfish moves only motivated by a transient improvement on their payoffs, without considering what will be their final payoffs when the game eventually reaches a pure Nash equilibrium. This observation naturally yields the question of whether agents taking decisions only based on what will be their short-term consequences, without considering what these decisions will cause tomorrow, might be considered as rational.

In [11], Bilò and Flammini propose a new definition of selfish agents by giving them a more farsighted view of their actions. An agent knows she is part of a multiplayer game and also knows that the game will not stop right after she has performed an improving step. Assume that in state σ player i has an improving step and that if she performs such a move a sequence of improvements begins leading the game toward a pure Nash equilibrium σ'. In this scenario, player i is mostly interested in comparing the payoff she is experiencing in σ with the one she can get at σ'. The idea is that, if $\omega_i(\sigma')$ is worse than $\omega_i(\sigma)$, player i is damaged by the consequences of her improving step; hence, she had better not performed it. When more than just one equilibrium can be reached from a particular state, by following a classical worst-case analysis, it can be assumed that the agent will compare the payoff in the current state with that at the equilibrium yielding the worst payoff for her. Such a viewpoint is clearly based upon the definition of a ground set of equilibria the agents will compare a generic state with. According to these comparisons, a possible nonempty set of new equilibria may arise and the process may be iterated recursively until a fixed point is reached and the final set of desired equilibria is created. Such a set is called the set of Second Order equilibria. Using different definitions of equilibrium for defining the ground set, it is possible to achieve different

sets of Second Order equilibria. [11] focuses on the definition and the evaluation of Second Order Nash equilibria, that is, with a ground set given by the set of pure Nash equilibria.

Consider the following generalization of the Nash dynamics graph $D(\mathscr{G})$. Given a set of (equilibria) states $E \subseteq S$, let $D(\mathscr{G}, E) = (N, A)$ be a directed graph in which $N = S$ and there exists an edge between σ and σ' if and only if there exists an improving step from σ to σ' and $\sigma \notin E$. Edges are still labeled with the index of the player performing the related improving step. Clearly, $D(\mathscr{G}, \emptyset)$ coincides with the Nash dynamics graph of $\mathscr{G}$. Define $\rho_E(\sigma)_i^k$ as the set of all the states of $\mathscr{G}$ that can be reached starting from σ by following a path of length at most k whose first arc is labeled with index i in the graph $D(\mathscr{G}, E)$. The set $\rho_E(\sigma)^k = \bigcup_{i=1}^n \rho_E(\sigma)_i^k$ will denote the set of all states that can be reached from σ by following a path of length at most k in $D(\mathscr{G}, E)$. When $E = \emptyset$, we will simply remove the subscript E from the notation. Let us also define $P(\sigma)$ as the set of players having an improving step in σ. Assume that the payoffs of each player are costs to be minimized. [11] introduces the following definition.

Definition 9.1. Let $\mathscr{G}$ be a game with the FIP property. The set $N^k(\mathscr{G}) = \{\sigma \in S : \forall i \in P(s) \text{ and } \forall \sigma' \in \rho(\sigma)_i^1, \exists \sigma'' \in N^k(\mathscr{G}) \text{ such that } \sigma'' \in \rho_{N^k(\mathscr{G})}(\sigma')^k \text{ and } \omega_i(\sigma) < \omega_i(\sigma'')\}$ is the set of all the Second Order k-Nash equilibria of $\mathscr{G}$, for any integer $k \geq 0$.

Intuitively, this rather involved definition says that a state σ is a Second Order k-Nash equilibrium, for some integer $k \geq 0$, if all the players that have an improving step in σ would experience a payoff worse than the one they get in σ in one of the Second Order k-Nash equilibria resulting from an evolutive process of at most k improving steps taking place after their first defection. Such a definition is clearly recursive. However, it can be shown that it is well posed, in the sense that it admits a unique set of solutions or fixed points. First of all, it is easy to see that $N^0(\mathscr{G})$ coincides with the set of pure Nash equilibria for $\mathscr{G}$ and that each pure Nash equilibrium is a Second Order k-Nash equilibrium, for any integer $k \geq 1$. Then, by proving that there exists a value k^* for which all the sets $N^k(\mathscr{G})$ become the same for any $k \geq k^*$, the following final definition can be given.

Definition 9.2. Let $\mathscr{G}$ be a game with the FIP property. Each state $\sigma \in N^{k^*}(\mathscr{G}) =_{def} N(\mathscr{G})$ is a Second Order Nash equilibrium.

Clearly, since $N^0(\mathscr{G}) \subseteq N^k(\mathscr{G})$ for any $k \geq 0$, we have that the price of stability of Second Order Nash equilibria is not worse than that of pure Nash equilibria, while the price of anarchy of pure Nash equilibria is not worse than that of Second Order Nash equilibria. The interested reader is referred to [11] for applications of Second Order Nash equilibria to some traditional games, as well as extensions and variations of this notion of equilibrium.

9.7 Coping with Incomplete Information

In highly decentralized and pervasive networks, the assumption that each agent knows the strategy adopted by any other agent may be too optimistic or even infeasible. In such situations, the set of agents of which each agent knows the chosen strategy is modeled by means of a social knowledge graph, that is, a directed graph $K = (P, E)$ whose set of nodes coincides with the set of players in the game, and there is a directed edge from player i to player j if and only if i knows the strategy adopted by j. Since i always knows her chosen strategy, it is assumed that any social knowledge graph contains all self-loops. For each game $\mathscr{G}$ and social knowledge graph K, a graphical game $(\mathscr{G}, K)$ is obtained by extending its definition in such a way that the payoff of each player can be influenced only by the strategies adopted by the adjacent ones. In the following, we discuss graphical linear congestion games and graphical Shapley cost sharing games which have been studied in [9] and [10] respectively.

Graphical Linear Congestion Games. In a graphical congestion game $(\mathscr{G}, K)$, the payoff of player i is defined as $\omega_i(\sigma) = \sum_{e \in \sigma_i} d_e(n_e^i(\sigma, K))$, where $n_e^i(\sigma, K) = |\{j \in P : e \in \sigma_j \text{ and } (i, j) \in E(K)\}|$ is the number of players using e in σ that are neighbors of i in K, including i herself.

The classical potential function defined by Rosenthal no longer applies to the graphical case; hence, as a first approach to the problem, a complete characterization of the cases possessing pure Nash equilibria and the FIP property needs to be achieved. The topology of K plays a fundamental role in this issue. In particular, if K is undirected, $(\mathscr{G}, K)$ is an exact potential game and thus isomorphic to a classical congestion game; if K is directed, an equilibrium is not guaranteed to exist in general, but $(\mathscr{G}, K)$ possesses the FIP property and an equilibrium can be found in polynomial time if K is acyclic, even if finding the best equilibrium remains an intractable problem.

Limited knowledge of the players yields two different possible definitions for the latencies experienced by a player in each state: The *presumed latency* is the one the player believes she suffers due to the fact that she is only aware of the existence of her neighbors, and is defined as $pres_i(\sigma) = \omega_i(\sigma)$. The *perceived latency* is the one she actually experiences due to all the players using the resource, and is defined as $perc_i(\sigma) = \sum_{e \in \sigma_i} d_e(n_e(\sigma))$. According to these definitions, four different social functions can be asked to be minimized, namely, the sum and the maximum of the presumed latencies and the sum and the maximum of perceived ones. Given a bound Δ on the maximum degree of K, for all the cases in which pure Nash equilibria are guaranteed to exist, tight lower and upper bounds on the price of stability and asymptotically tight bounds on the price of anarchy of pure strategies have been achieved for such social functions. These results have been also extended to load balancing games and are summarized in Table 9.1.

Graphical Shapley Cost Sharing Games. In a graphical Shapley cost sharing game $(\mathscr{G}, K)$, the payoff of player i is defined as $\omega_i(\sigma) = \sum_{e \in \sigma_i} \frac{c_e}{n_e^i(\sigma, K)}$. Again, because

Table 9.1 Summary of the results on the price of anarchy and stability with respect to the presumed and perceived social cost functions

(a) Presumed Latency

Presumed Latency	PoS^{sum}, PoS^{max}	PoA^{sum}, PoA^{max}
Undirected graph	$2, \Theta(\Delta+1)$	$\Theta(\Delta+1), \Delta+1$
Acyclic DAG	$\Theta(\Delta+1), \Delta+1$	$\Theta(\Delta+1), \Delta+1$

(b) Perceived Latency in Congestion Games

Perceived Latency	Congestion Games	
	PoS^{sum}, PoS^{max}	PoA^{sum}, PoA^{max}
Undirected graph	$n, n \div n\sqrt{\Delta+1}$	$\Theta(n(\Delta+1))$
Acyclic DAG	$\Theta(n(\Delta+1))$	$\Theta(n(\Delta+1))$

(c) Perceived Latency in Load Balancing Games

Perceived Latency	Load Balancing Games	
	PoS^{sum}, PoS^{max}	PoA^{sum}, PoA^{max}
Undirected graph	$n, \Theta(n)$	$\Theta(n)$
Acyclic DAG	$\Theta(n)$	$\Theta(n)$

of the fact that some receivers can be hidden to other ones, graphical Shapley cost sharing games can no longer be isomorphic to congestion games when considering the presence of social knowledge graphs.

If K is a directed acyclic graph (DAG), the same technique used for graphical linear congestion games shows that $(\mathscr{G}, K)$ possesses the FIP property and that an equilibrium can be computed in polynomial time. If K is either undirected or directed cyclic, existence of pure Nash equilibria is no longer guaranteed even for the multicast case. However, when K is undirected, the restriction to the load balancing case can be shown to be isomorphic to general potential games, and hence to have the FIP property. This does not hold when K is a directed graph containing cycles.

The results on the price of anarchy and stability which can be achieved on the multicast case are quite surprising. The price of stability achievable by any DAG is at least $\frac{1}{2}\log n$, while the price of anarchy for complete DAGs can be shown to be at most $\log^2 n$. This result can be achieved by proving that the set of Nash equilibria induced by any complete DAG K^* on any instance I coincides with the set of solutions obtained after a first round of best responses in which the receivers enter sequentially the game I starting from the empty configuration according to their topological ordering in K^*. The upper bound on the price of stability follows by exploiting a result presented in [18]. Putting everything together, we have that the complete DAG, if used as a universal knowledge graph, is able to contain the price of anarchy of the graphical Shapley multicast cost sharing game under a polylogarithmic bound.

When a particular instance of the Shapley multicast cost sharing game is fixed in advance, we have that the price of stability achieved by any DAG must be at least $\frac{4n}{n+3}$. On the other side, it is possible to prove that for any instance I there always exists a DAG $K(I)$ achieving a price of anarchy of at most $\frac{4n}{n+3}$ if $n = 2, 3$ and of

at most $\frac{4(n-1)}{n+1}$ if $n \geq 4$, hence obtaining an upper bound on the price of anarchy almost matching the lower bound on the price of stability achievable by any DAG. Unfortunately, it is not known how to construct efficiently the graph $K(I)$. However, given any r-approximation of the optimal multicasting tree, it is possible to compute in polynomial time (by using a simple depth-first search) a DAG achieving a price of anarchy of at most $\frac{4n}{n+3}r$ if $n = 2, 3$ and of at most $\frac{4(n-1)}{n+1}r$ if $n \geq 4$, lowering the price of anarchy of this game to a constant value.

These achievements are twofold: from one side, they shed some light on how the lack of knowledge among players can have impact on the total cost of the self-emerging networks created by the interactions of selfish users; from the other side, they show that the idea of hiding some players to others is a powerful instrument that a central authority managing the game can use in order to obtain solutions whose cost may be not too far from the optimal one without interfering directly on the choices made by the players. This situation can be seen as another evidence of the famous Braess' paradox [13]. According to this paradox, there are cases in which adding fast links in a network results in a decrease of performance or, symmetrically, hiding some fast links from the players can improve the network performance. This naturally extends to graphical Shapley multicast cost sharing games where hiding some of the players to other ones can yield better solutions. In this sense, the less players know, the most they are "cooperative."

9.8 Open Problems and Future Research

Algorithmic Game Theory is a relatively young research area. As a consequence, it provides a wide landscape for setting new ideas, theories, and applications. Also, lots of questions have been raised by the research conducted so far. In this section we propose a list of the most important and interesting open problems related to the topics covered in the chapter.

Congestion games. The price of anarchy and stability of linear congestion games have been exactly estimated both in the general case and in the load balancing case. Much of the current trend in the study of congestion games is thus moving towards the characterization of more general cases allowing more complicated delay functions as well as different demands; see, for instance, [1, 30]. Most of these results are concerned with the price of anarchy when considering, as a social function, the sum of the latencies experienced by all the players, while the study of the price of stability and the analysis of the social function given by the maximum delay experienced by the players still deserves further investigation. Also, problems related to the existence of pure Nash equilibria, the speed of convergence towards Nash equilibria, and the complexity of computing Nash equilibria in various special cases deserve further investigation.

Multicast cost sharing games. This is a widely open research topic since one can define several reasonable socioeconomic properties a cost sharing method should satisfy. A first effort in this direction can be found in [7, 20]. The latter paper, in particular, provides a complete characterization of the price of anarchy which can be achieved by cost sharing methods satisfying certain desiderata. Such properties are rather strong and a similar study based on weaker constraints like the ones defined in [7] could be an interesting research direction. Moreover, fewer results are known regarding the price of stability. In this particular setting, the major open problem regards the characterization of the price of stability of the Shapley value method in undirected networks, which, in spite of a constant lower bound, has been upper-bounded by $\mathcal{O}(\log n)$. A sublogarithmic upper bound is only known for the special case of broadcasting games [28].

Communication games in all-optical networks. While the complete information level has been fully understood, the main question left open under the lower levels is the determination of functions that yield Nash equilibria on every topology with performance better than the worst possible one assigning a different color to each agent. Moreover, under incomplete information, it would be interesting to improve and extend the results on specific topologies. In this latter case, moreover, nice connections between payment functions and online algorithms, allowing us to cope with the arbitrary order of the moves of the agents, have been shown. It would be challenging to understand the conditions and eventual systematic methods that could yield converging payment functions preserving online algorithm performance under incomplete information.

Second-order Nash equilibria. A lot of open questions have been introduced by considering the extended notion of rationality of selfish players. The first one is certainly that of giving further validation to Second Order equilibria by using them in conjunction with other known equilibria notions and presenting good applications. To this end, the definition of Second Order Sink equilibria seems to be a promising research direction. Moreover, there is the important issue of understanding the power of different ordering strategies in influencing the performance of these equilibria. An interesting question can be also that of trying to understand if the use of Second Order equilibria can lead sequences of improving steps towards better states. In [11] only impatient prudent agents have been considered. A final open issue is certainly that of analyzing the other three possible definitions for rational agents as well as the Second Order equilibria yielded by rush agents.

Coping with incomplete information. Possible applications of social knowledge graphs include the design of protocols and P2P systems which limit the visibility of the other peers, or simply, at a more foundational level, the possibility of using them as an intermediate methodological tool for defining cost shares and payoffs so as to induce good overall performance without direct interference in user decisions.

For Graphical linear congestion games, a crucial observation is that better bounds with respect to the ones reported in Table 9.1 can be obtained for specific social

graphs. In fact, for the undirected complete graph, constant bounds are derived directly from the classical linear congestion game. Many questions are left open. Besides tightening the various constant multiplicative gaps, it would be worth closing the gap between the upper and the lower bound on the price of stability for the maximum perceived latency social function. The investigation of the consequences of a minimum degree very close to n is another interesting issue. Moreover, what about nonlinear latency functions? While the convergence for directed acyclic graphs works for all the latency functions, in the undirected case, convergence strictly relies on linearity. It would be also worth investigating the expected price of stability and anarchy when the knowledge graph obeys some social behavior, as in Kleinberg's Small Word model [33, 34]. In general, are there universal bounded degree social graphs always guaranteeing good performance?

For Graphical multicast cost sharing games, besides closing the gaps between the upper and lower bounds on the price of anarchy and stability (such as the gap for the price of anarchy in the universal case), it would be interesting to extend the study to other graphical cost sharing games, like cost sharing congestion games. Finally, what about the effect of social graphs on the speed of convergence, that is, on the number of selfish moves needed to reach equilibria or on the performance achieved after a limited number of steps?

References

1. Aland, S., Dumrauf, D., Gairing, M., Monien, B., Schoppmann, F.: Exact price of anarchy for polynomial congestion games. In: Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS '06), LNCS 3884, pp. 218-229, 2006
2. Anshelevich, E., Dasgupta, A., Kleinberg, J. M., Tardos, E., Wexler, T., Roughgarden, T.: The price of stability for network design with fair cost allocation. In: Proceedings of the 45th Annual Symposium on Foundations of Computer Science (FOCS '04), IEEE Computer Society, pp. 295-304, 2004
3. Anshelevich, E., Dasgupta, A., Tardos, E., Wexler, T.: Near-optimal network design with selfish agents. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC '03). ACM Press, pp. 511-520, 2003
4. Awerbuch, B., Azar, Y., Epstein, A.: The price of routing unsplittable flow. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05), ACM Press, pp. 57-66, 2005
5. Azar Y., Epstein, A.: The hardness of network design for unsplittable flow with selfish users. In: Proceedings of the 3rd Workshop on Approximation and Online Algorithms (WAOA '05), LNCS 3879, pp. 41-54, 2005
6. Azar, Y., Jain, K., Mirrokni, V. S.: (Almost) optimal coordination mechanisms for unrelated machine scheduling. In: Proceedings of the 19th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA '08), pp. 323-332, 2008

7. Bilò, V.: The price of Nash equilibria in multicast transmission games. In: Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC '07), LNCS 4835, pp. 390-401, 2007

8. Bilò, V., Fanelli, A., Flammini, M., Melideo, G., Moscardelli, L.: Designing fast converging cost sharing methods for multicast transmissions. Theory of Computing Systems, online first, 2009

9. Bilò, V., Fanelli, A., Flammini, M., Moscardelli, L.: Graphical congestion games with linear latencies. In: Proceedings of the 20th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '08), pp. 194-196, 2008

10. Bilò, V., Fanelli, A., Flammini, M., Moscardelli, L.: When ignorance helps: graphical multicast cost sharing games. In: Proceedings of the 32nd International Symposium on Mathematical Foundations of Computer Science (MFCS '08), LNCS 5162, pp. 108-119, 2008

11. Bilò, V., Flammini, M.: Extending the notion of rationality of selfish agents: second order Nash equilibria. In: Proceedings of the 32nd International Symposium on Mathematical Foundations of Computer Science (MFCS '07), LNCS 4708, pp. 621-632, 2007

12. Bilò, V., Flammini, M., Moscardelli, L.: On Nash equilibria in non-cooperative all-optical networks. In: Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS '05), LNCS 3404, pp. 448-459, 2005

13. Braess, D.: Über ein Paradoxon der Verkehrsplanung. Unternehmensforschung, 12:258-268, 1968

14. Caragiannis, I.: Efficient coordination mechanisms for unrelated machine scheduling. In: Proceedings of the 20th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA '09), pp. 815–824, 2009

15. Caragiannis, I., Flammini, M., Kaklamanis, C., Kanellopoulos, P., and Moscardelli, L.: Tight bounds for selfish and greedy load balancing. In: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP '06), LNCS 4051, Part I, pp. 311-322, 2006

16. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Taxes for linear atomic congestion games. In: Proceedings of the 14th Annual European Symposium on Algorithms (ESA '06), LNCS 4168, pp. 184-195, 2006

17. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Improving the efficiency of load balancing games through taxes. In: Proceedings of the 4th International Workshop on Internet and Network Economics (WINE '08), LNCS 5385, pp. 374-385, 2008

18. Charikar, M., Mattieu, C., Karloff, H., Naor, J., Saks, M.: Best response dynamics in multicast cost sharing. In: Proceedings of the 20th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '08), pp. 70-76, 2008

19. Chekuri, C., Chuzhoy, J., Lewin-Eytan, L., Naor, J., Orda, A.: Non-cooperative multicast and facility location games. In: Proceedings of the 7th ACM Conference on Electronic Commerce (EC '06), ACM Press, pp. 72-81, 2006

20. Chen, H., Roughgarden, T., Valiant, G.: Designing networks with good equilibria. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '08), ACM Press, pp. 854-863, 2008

21. Christodoulou, G., Koutsoupias, E.: The price of anarchy of finite congestion games. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05), ACM Press, pp. 67-73, 2005

22. Christodoulou, G., Koutsoupias, E.: On the price of anarchy and stability of correlated equilibria of linear congestion games. In Proceedongs of the 13th Annual European Symposium on Algorithms (ESA '05), LNCS 3669, pp. 59-70, 2005

23. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination mechanisms. In: Proceedings of the 31st Annual International Colloquium on Automata, Languages, and Programming (ICALP '04), LNCS 3142, pp. 345-357, 2004

24. Christodoulou, G., Mirrokni, V., Sidiropoulos, A.: Convergence and approximation in potential games. In: Proceedings of the 23rd Symposium on Theoretical Aspects of Computer Science (STACS '06), LNCS 3884, Springer, pp. 349-260, 2006

25. Cole, R., Dodis, Y., Roughgarden, T.: Pricing network edges for heterogeneous selfish users. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC '03), ACM Press, pp. 521-530, 2003

26. Fanelli, A., Flammini, M., Moscardelli, L.: On the convergence of multicast games in directed networks. In: Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '07), ACM Press, pp. 330-338, 2007

27. Fanelli, A., Flammini, M., Moscardelli, L.: The speed of convergence in congestion games under best-response dynamics. In: Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP '08), LNCS 5125. Part I, pp. 796-807, 2008

28. Fiat, A., Kaplan, H., Levy, M., Olonetsky, S., Shabo, R.: On the price of stability for designing undirected networks with fair cost allocations. In: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP '06), LNCS 4051, Part I, pp. 608-618, 2006

29. Fleischer, L., Jain, K., Mahdian, M.: Tolls for heterogeneous selfish users in multicommodity networks and generalized congestion games. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS '04), IEEE Computer Society, pp. 277-285, 2004

30. Fotakis, D., Kontogiannis, S., Spirakis, P.: Selfish unsplittable flows. Theoretical Computer Science, 348(2-3): 226-239, 2005

31. Immorlica, N., Li, L., Mirrokni, V. S., Schulz, A.: Coordination mechanisms for selfish scheduling. In: Proceedings of the 1st Workshop on Internet and Network Economics (WINE '05), LNCS 3828, pp. 55-69, 2005

32. Karakostas, G., Kolliopoulos, S.: Edge pricing of multicommodity networks for heterogeneous selfish users. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS '04), IEEE Computer Society, pp. 268-276, 2004

33. Kleinberg, J.: The small-world phenomenon: an algorithm perspective. In: Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC '00), ACM Press, pp. 163-170, 2000

34. Kleinberg, J.: Small-world phenomena and the dynamics of information. In: Proceedings of the 14th Advances in Neural Information Processing Systems (NIPS '01), pp. 431-438, 2001

35. Koutsoupias, E., Papadimitriou, C. H.: Worst-case Equilibria. In: Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS '99), LNCS 1653, pp. 404-413, 1999

36. Mirrokni, V. S., Vetta, A.: Convergence issues in competitive games. In: Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX '04), LNCS 3122, pp. 183-194, 2004

37. Monderer, D., Shapley, L. S.: Potential games. Games and Economic Behavior. 14:124-143, 1996

38. Nash, J.: Equilibrium points in n-person games. In: Proceedings of the National Academy of Sciences, volume 36, pp. 48-49, 1950

39. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. V.: Algorithmic game theory. Cambridge University Press, 2007

40. Pigou, A. C.: The economics of welfare. Macmillan, 1920

41. Rosenthal, R. W.: A class of games possessing pure-strategy Nash equilibria. International Journal of Game Theory, 2:65-67, 1973

42. Roughgarden, T.: Stackelberg scheduling strategies. SIAM Journal on Computing, 33(2): 332-350, 2004

43. Roughgarden, T.: On the severity of Braess's Paradox: designing networks for selfish users is hard. Journal of Computer and System Sciences, 72(5): 922-953, 2006

44. Roughgarden, T., Tardos, E.: How bad is selfish routing? Journal of the ACM, 49(2): 236-259, 2002

45. Shapley, L. S.: The value of n-person games. Contributions to the theory of games, pp. 31-40, Princeton University Press, 1953

46. Suri, S., Tóth, C., Zhou, Y.: Selfish load balancing and atomic congestion games. Algorithmica, 47(1): 79-96, 2007

Chapter 10
Permutation Routing and (ℓ,k)-Routing on Plane Grids

Ignasi Sau and Janez Žerovnik

Abstract The packet routing problem plays an essential role in communication networks. It consists in transferring data from some origins to some destinations within a reasonable amount of time. In the (ℓ,k)-routing problem, each node can send at most ℓ packets and receive at most k packets. Permutation routing is the particular case $\ell = k = 1$. In the r-central routing problem, all nodes at distance at most r from a fixed node v want to send a packet to v. Here, we survey the results on permutation routing, the r-central routing, and the general (ℓ,k)-routing problems on regular plane grids, that is, square grids, triangular grids, and hexagonal grids. We assume the *store-and-forward Δ-port model* with synchronous transmission, and we consider both full and half-duplex networks.

Key words: packet routing, distributed algorithm, (ℓ,k)-routing, plane grids, permutation routing, shortest path, oblivious algorithm

10.1 Introduction

In telecommunication networks, it is essential to be able to route communications as quickly as possible. In this context, the *packet routing* problem plays a capital role. In this problem we are given a network and a set of packets to be routed through the nodes and the edges of the network graph. A packet is characterized by an origin and

Ignasi Sau

MASCOTTE, INRIA, I3S, CNRS UMR6070, University of Nice-Sophia Antipolis, France, and Graph Theory and Combinatorics Group, Department of Applied Mathematics IV, Universitat Politècnica de Catalunya, Barcelona, Spain, e-mail: `Ignasi.Sau@sophia.inria.fr`

Janez Žerovnik

University of Maribor, Smetanova 17, Maribor 2000, Slovenia, and Institute of Mathematics, Physics and Mechanics, Jadranska 19, Ljubljana, Slovenia, e-mail: `janez.zerovnik@imfm.uni-lj.si`

a destination node, and typically an edge can be used by no more than one packet at a time. The objective is to find an algorithm to compute a schedule to route all packets which minimizes the total delivery time. This problem has been widely studied in the literature under many different assumptions. In 1988, Leighton, Maggs, and Rao proved in [27] (see also [25]) that there exists a schedule for routing any set of packets with edge-simple paths on a general network, in optimal time of $\mathcal{O}(C+D)$ steps, where C is the congestion (maximum number of paths sharing an edge) and D the dilation (length of a longest path), assuming that the paths are given a priori. The proof used the Lovász Local Lemma and was nonconstructive. This result was further improved in [24], where the same authors gave an explicit algorithm, using Beck's constructive version of the Local Lemma. These algorithms to compute the optimal schedule are centralized. Then, in [34] Ostrovsky and Rabani gave a distributed randomized algorithm running in $\mathcal{O}(C+D+\log^{1+\varepsilon}(n))$ steps. We give a more detailed overview of these results in Section 10.1.1.

Although these results are asymptotically tight, they deal with a general network, and in many cases it is possible to design more efficient algorithms by looking at specific packet configurations or network topologies. For instance, it is natural to bound the maximum number of messages that a node can send or receive. We focus on this point in Section 10.1.2, where we will formally define the problem considered here. On the other hand, the structure of the network under study plays a major role in the quality and the simplicity of the solution. For example, in a radio wireless environment, cellular networks are usually modeled by a *hexagonal tessellation* (or *hexagonal grid*), where centers of cells represent base stations. The cells of the hexagonal tessellation have good diameter to area ratio and still have a simple structure. If centers of neighboring cells are connected, the resulting structure is called a *triangular grid*. Notice that hexagonal grids are subgraphs of the triangular grid. We will talk about such networks in Section 10.1.3. In this survey we focus on the study of the (ℓ,k)-routing problem in convex subgraphs of the square, triangular, and hexagonal grids. A graph is called convex if it contains all shortest paths between any pair of vertices.

10.1.1 General Results on Packet Routing

In this section we provide a fast overview of the state of the art of the general packet routing problem, in both the off-line and online settings, focusing mostly on the latter. We begin by recalling three classical lower bounds for the packet routing problem.

10.1.1.1 Classical Lower Bounds

In the packet routing problem, there are three classical types of lower bounds for the running time of any algorithm:

1. Distance bound. The longest distance over the paths of all packets (usually called *dilation*) constitutes a lower bound on the number of steps required to route all the packets.
2. Congestion bound. The *congestion* of an edge of the network is defined as the number of paths using this edge. Then, the greatest congestion over all the edges of the network is also a lower bound on the number of steps, since at each step an edge can be used by at most one packet.
3. Bisection bound. Let $G = (V, E)$ be the graph which models the network, and $C \subseteq E$ a cut-set dissecting G into two components G_1 and G_2. Let m be the number of packets with origin in G_1 and destination in G_2. Then, the number of routing steps used by any algorithm will be at least $\left\lceil \frac{m}{|C|} \right\rceil$.

10.1.1.2 Off-line Routing

Given a set of packets to be sent through a network, a *path system* is defined as the union of the paths that each packet must follow. For a general network and any set of n demands, we have seen in Section 10.1.1.1 that the dilation and the congestion provide two lower bounds for the routing time. This proves that the *dilation* + *congestion* of a path system used for the routing procedure is a lower bound of twice the routing time. In a celebrated paper, Leighton, Maggs, and Rao proved the following theorem:

Theorem 10.1 ([27]). *For any set of requests and a path system for these requests, there is an* off-line *routing protocol that needs $\mathcal{O}(C + D)$ steps to route all the requests, where C is the congestion and D is the dilation of the path system.*

In addition, in [45] the authors show that, given the set of packets to be sent, it is possible to find in polynomial time a path system with the value of $C + D$ within a factor of 4 of the optimum. Thus, Theorem 10.1 can be phrased in a more general way:

Theorem 10.2 ([45]). *For any set of requests, there is an* off-line *routing protocol that needs $\mathcal{O}(C + D)$ steps to route all the requests, where $C + D$ is the minimum congestion + dilation over all possible path systems.*

Furthermore, this routing protocol uses fixed buffer size, i.e., the queue size at all nodes is bounded by a constant at each step. Nevertheless, it is important to notice that a huge constant may be hidden inside the $\mathcal{O}$ notation. As we have said in the introduction, this result was further improved in [24], where the same authors gave an explicit algorithm. These algorithms to compute the optimal schedule are centralized. In a distributed algorithm nodes must make their decisions independently, based on the packets they see, without the use of a centralized scheduler. Then, in [34] Ostrovsky and Rabani give a distributed randomized algorithm running in $\mathcal{O}(C + D + \log^{1+\varepsilon}(n))$ steps.

We refer to Scheideler's thesis [41] for a complete compilation of general packet routing algorithms.

10.1.1.3 Online Routing

In the online setting, the oldest online protocol that deviates only by a factor logarithmic in n path collections is the protocol presented by Leighton, Maggs and Rao [27], running in $\mathcal{O}(C + D\log(Dn))$ steps with high probability. The authors call the algorithm online, rather than distributed. This schedule assumes that the paths are given a priori; hence, it does not focus on the problem of choosing the paths to route the packets.

The results of [2] provide a routing algorithm that is $\log n$-competitive with respect to the congestion. In other words, it is worse than an optimal off-line algorithm only by a factor $\log n$. In this setting the demands arrive one by one and the algorithm routes calls based on the current congestion on the various links in the network, so this can be achieved only via centralized control and serializing the routing requests. In [4] the authors gave a distributed algorithm that repeatedly scans the network so as to choose the routes. This algorithm requires shared variables on the edges of the network and hence is hard to implement. Note that the two online algorithms above depend on the demands and are therefore *adaptive*. An *oblivious* routing strategy is specified by a path system $\mathscr{P}$ and a function w assigning a weight to every path in $\mathscr{P}$. This function w has the property that for every source-destination pair (s,t), the system of flow paths $\mathscr{P}_{s,t}$ for (s,t) fulfills $\sum_{q \in \mathscr{P}_{s,t}} w(q) = 1$. One can think of this function as a frequency distribution among several paths going from an origin s to a destination t. In *adaptive* routing, however, the path taken by a packet may also depend on other packets or events taking place in the network during its travel. Note that every oblivious routing strategy is obviously online and distributed.

The first paper to perform a worst-case theoretical analysis of oblivious routing is the paper of Valiant and Brebner [50], who considered routing on specific network topologies such as the hypercube. They gave a randomized oblivious routing algorithm. Borodin and Hopcroft [6] and subsequently [19] have shown that deterministic oblivious routing algorithms cannot approximate well the minimal load on any nontrivial network.

In a recent paper, Räcke [37] gave the construction of a polylogarithmic competitive oblivious routing algorithm for general undirected networks. It seems truly surprising that one can come close to minimal congestion without any information on the current load in the network. This result has been improved in [5]. Lower bounds on the competitive ratio of oblivious routing have been studied for various types of networks. For example, for the d-dimensional mesh, Maggs et al. [36] gave an $\omega(\frac{C_*}{d}(\log n))$ lower bound on the competitive ratio of an oblivious algorithm, where C_* is the optimal congestion.

So far, the oblivious algorithms studied in the literature have focused on minimizing the congestion while ignoring the dilation. In fact, the quality of the paths should be determined by the congestion C and the dilation D. A fundamental question is whether C and D can be controlled simultaneously. An appropriate parameter to capture how good the dilation of a path system is is the *stretch*, defined as the maximum over all packets of the ratio between the length of the path taken by the routing protocol and the length of a shortest path from source to destination. In a re-

cent work, Bush et al. [8] considered again the case of the d-dimensional mesh. They presented an online algorithm for which C and D are both within $\mathcal{O}(d^2)$ of the potential optimum, i.e., $D = \mathcal{O}(d^2 D_*)$ and $C = \mathcal{O}(dC_* \log(n))$, where D_* is the optimal dilation (note that by [36], it is impossible to have a factor better than $\omega(\frac{C_*}{d}(\log n))$).

There is a simple counter-example network which shows that in general the two metrics (*dilation* and *congestion*) are orthogonal to each other: take an adjacent pair of vertices u, v, and $\Theta(\sqrt{n})$ disjoint paths of length $\Theta(\sqrt{n})$ between u and v. For packets traveling from u to v, any routing algorithm that minimizes congestion has to use all the paths; however, in this way some packets follow long paths (i.e., not using the edge between u and v), giving high stretch. Nevertheless, in grids [8] and in some special kinds of geometric networks [7] the congestion is within a polylogarithmic factor from optimum and stretch is constant (d being the dimension). As mentioned before, an interesting open problem is to find other classes of networks where the congestion and stretch are minimized simultaneously [3]. Possible candidates for such networks could be bounded-growth networks, or networks whose nodes are uniformly distributed in closed polygons, which describe interesting cases of wireless networks.

The recent paper of Maggs [30] surveys a collection of theoretical results that relate the congestion and dilation of the paths taken by a set of packets in a network to the time required for their delivery.

10.1.2 Routing Problems

The initial and final positioning of the packets has a direct influence on the time needed for their routing. Considering static packet configuration, the most studied constraints refer to the maximum number of packets that a node can send and receive. Due to their practical importance, some of these problems have specific names:

1. Permutation routing. Each node is the origin and the destination of at most one packet. To measure the routing capability of an interconnection network, the partial permutation routing (PPR) problem is usually used as the metric.
2. (ℓ, k)-routing. Each node is the origin of at most ℓ packets and destination of at most k packets. Permutation routing corresponds to the case $\ell = k = 1$ of (ℓ, k)-routing. Another important particular case is the $(1, k)$-*routing*, in which each node can send at most one packet and receive at most k packets.
3. $(1, any)$-routing. Each node is the origin of at most one packet but there are no constraints on the number of packets that a node can receive.
4. r-central routing. All nodes at distance at most r of a central node send one message to this central node. This pattern is very close to the *broadcast* pattern.

In all these problems we are given an initial packet configuration, and the objective is to route all packets to their respective destinations minimizing the total routing

time, under the constraint that each edge can be used by at most one packet at the same time.

Besides the constraints about the initial and final positions of the packets, there also exist different routing models at the intermediate nodes of the network. For instance, in the *hot potato model* no packet can be stored at the nodes of the network, whereas in the *store-and-forward* at each step a packet can either stay at a node or move to an adjacent node. Another widely used model is *wormhole* routing [26].

On the other hand, one can consider constraints on the number of incident edges that each node of the network can use to send or receive packets at the same time. In the Δ-port model [11], each node can send or receive packets through all its incident edges at the same time. Here we study the store-and-forward Δ-port model. In addition, we suppose that cohabitation of multiple packets at the same node is allowed. Thus, a queue is required for each outgoing edge at each node. We also suppose that packets move in a synchronous way and that it takes exactly one time unit for a packet to traverse a link.

The nature of the links of the network is another factor that influences the routing efficiency. The type of link is usually one of the following: *full-duplex* or *half-duplex*. In the full-duplex case there are two links between two adjacent nodes, one in each direction. Hence two packets can move, one in each direction, simultaneously. In the half-duplex case only one packet can move between two nodes, in either direction of the edge. In this survey we consider both half- and full-duplex links.

10.1.3 Topologies

We now give a brief summary of various cases of (ℓ, k)-routing and $(1, any)$-routing that have been studied for several specific topologies. More precisely, we first list the most important results for some networks which have attracted a great interest in the literature, such as hypercubes and circulant graphs. Then we move to plane grids. It is well known that there exist only three possible tessellations of the plane into regular polygons [32]: squares, triangles, and hexagons. These graphs are those which we study in this survey.

10.1.3.1 Different Network Topologies

In [14] the authors studied the permutation routing problem in low-dimensional hypercubes ($d \leq 12$). They gave optimal or "good in the worst case" oblivious algorithms, i.e., algorithms in which the path used by a packet is entirely determined by its origin and its destination. Another network widely studied in the literature is the two-dimensional mesh with row and column buses (with point-to-point communication). This network can also be diversified according to the capacities of the buses. In [47] Suel gave a deterministic algorithm to solve the permutation routing problem

in such networks. It gives a schedule using at most $n + o(n)$ steps and a queue of size 2, where the queue is the maximum number of packets that have to be stored at an intermediate node. He also proposed a deterministic algorithm for r-dimensional arrays with buses working in $(2 - \frac{1}{r})n + o(n)$ steps and still using queues of size 2. In [22], the authors studied the (ℓ, ℓ)-routing problem in the mesh grid with two diagonals and gave, for $\ell \geq 9$, a deterministic algorithm using $\frac{2\ell n}{9} + (\ell n^{2/3})$ steps.

In [13], the authors introduced an algorithm called *big foot* algorithm. The idea of this algorithm is to identify two types of links and to move towards the destination using first the links of the first type and then those of the second type. The algorithms we develop will use such a strategy. Hwang, Lin, and Jan [13] gave an optimal algorithm for the permutation routing problem in full-duplex 2-circulant graphs. The same algorithm is optimal for double-loop networks, i.e., oriented 2-circulant graphs. Recall that a circulant graph is a graph on n vertices in which the ith vertex is adjacent to the $(i + j)$th and $(i - j)$th vertices (mod n) for each j in a list l of positive integers.

Another network of great practical importance is the *double-loop* network: a network modeled by a graph $G = (V, E)$ with $V = \{v_0, \ldots, v_{n-1}\}$ such that there are two integers h_1 and h_2 such that $E = \{v_i v_{i \pm h_1}, v_i v_{i \pm h_2} : i = 0, \ldots, n-1\}$ (the indices being taken modulo n). The permutation routing problem in this network is studied in [9]. The authors gave an algorithm for the permutation routing problem which on average uses 1.12ℓ steps (the expected value being empirically measured). In [10], an optimal centralized permutation routing algorithm in k-circulant graphs ($k \geq 2$) is given (with polylogarithmic time complexity for $k = 2$).

The problem has been also studied for packets arriving dynamically. In [12], the author gave an optimal online schedule for the linear array. He also gave a 2-approximation for rings and showed that, using shortest path routing, no better approximation algorithm exists. In [18], the authors studied Cube Connected Cycles: $CCC(n, 2^n)$ (hypercubes of dimension n where each vertex is replaced by a cycle of length n). They gave an algorithm working in $\mathcal{O}(n^2)$ with $\mathcal{O}(1)$ buffers for the online partial permutation routing (PPR).

10.1.3.2 Plane Grids

Maybe the most studied networks in the literature are the two-dimensional grids (or plane grids), and among them in particular the $n \times n = N$ square grid has deserved special attention. Let us briefly overview what has been previously done on (ℓ, k)-routing in plane grids.

In [28] the first running time $2n - 2$), and queues of size 1,008 appears. The queue size is reduced in [38] to 112 and further in [44] to 81. Furthermore, in [44] the authors provide another algorithm running in near-optimal time $2n + \mathcal{O}(1)$ steps with a maximum queue size of only 12. [31] gives an *asymptotically* optimal algorithm for $(1, k)$-routing on plane grids, with queues of small constant size. They introduced for the first time the $(1, k)$-routing and the $(1, any)$-routing problems. This result was further improved in [43], where the authors gave a near-optimal deterministic algo-
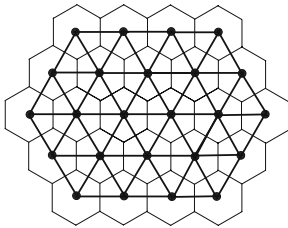
Fig. 10.1 Hexagonal network ($\triangle$) and hexagonal tessellation ($\bigcirc$)

rithm running in $\sqrt{k}\frac{n}{2} + \mathcal{O}(n)$ steps. Another algorithm was given that is slightly worse in terms of number of steps, but with queues of size only 3. In the same paper, lower bounds and near-optimal randomized and deterministic algorithms were proposed for the general problem of (ℓ,k)-routing on square grids. The algorithms were also extended to higher dimensional meshes which performed (ℓ,ℓ)-routing in $\mathcal{O}(\ell n)$ steps, the lower bound being $\Omega(\sqrt{\ell kn})$ for (ℓ,k)-routing [43]. Finally, in [35], the authors gave deterministic and randomized algorithms for (ℓ,k)-routing on square grids, with constant queue size. The running time is $\mathcal{O}(\sqrt{\ell kn})$ steps, which is optimal according to the bound of [43]. This work closed a gap in the literature, since optimal algorithms were only known for $\ell = 1$ and $\ell = k$. Oblivious permutation routing algorithms were given by Iwama and Miyano [16, 17] and Litman and Moran-Schein [29]. For generalization to d-dimensional meshes and (k,k)-routing see [15, 23, 33]. On triangular and hexagonal grids, the best results are randomized algorithms with good performance [42].

Nodes on a hexagonal network are placed at the vertices of a regular triangular tessellation, so that each node has up to six neighbors. These networks have been studied in a variety of contexts, especially in wireless and interconnection networks. The most known application may be to model cellular networks with hexagonal networks where nodes are base stations. Furthermore, these networks have been also applied in chemistry to model benzenoid hydrocarbons [20, 48], and in image processing and computer graphics [21].

In a radiocommunication wireless environment [32], the interconnection network among base stations constitutes a hexagonal network, i.e., a triangular grid, as it is shown in Figure 10.1.

Tessellation of the plane with hexagons may be considered as the most natural because cells have optimal diameter to area ratio (in the sense that this ratio is greater than in square or triangular grids). Hexagonal networks are finite subgraphs of the triangular grid. The triangular grid can also be obtained from the basic 4-mesh by adding NE to SW edges, which is called a 6-mesh in [49]. Here we study convex subgraphs of the square, triangular, and hexagonal grids.

In the next section we briefly outline some results on the permutation routing problem on hexagonal grids. Some details are given in order to illustrate the main ideas that can be applied to other grids with some natural modifications. We use the

store-and-forward Δ-port model, and consider both full- and half-duplex networks. Recall that here we assume there are no bounds on the queue size.[1]

10.2 Optimal Permutation Routing Algorithm

10.2.1 Preliminaries

Nodes on a hexagonal network are placed at the vertices of a regular triangular tessellation, so that each node has up to six neighbors. We deal in this section with a hexagonal mesh network $G = (V,E)$ with full-duplex links, that is, an edge of the network can be crossed by two simultaneous messages, one in each direction. Equivalently, each edge between two vertices u and v is made of two independent arcs (u,v) and (v,u), as illustrated in Figure 10.2a.

a)   b)
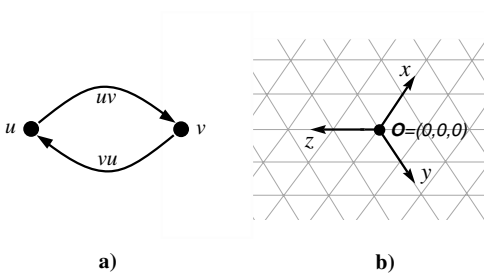
Fig. 10.2 a) Each edge consists of two independent links. b) Axis used in a hexagonal network

In [32] the authors solved the problem of routing a *single* message through a hexagonal communication network. The idea (first introduced in [46]) that will be very useful later is the representation of any address in a basis consisting of three unitary vectors $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ on the directions of the three axes x, y, z with a $120°$ angle between them, intersecting on an arbitrary (but fixed) node O labeled with the address $\boldsymbol{O} = (0,0,0)$, as depicted in Figure 10.2b.

Thus, assume that each vertex $P \in V$ is labeled with an address $\boldsymbol{P} = (P_1, P_2, P_3)$ expressed in this basis $\{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$ with respect to the origin O. At the beginning, each vertex S knows the address of the destination vertex D of the message placed initially at S, and computes the relative address $\overrightarrow{SD} = \boldsymbol{D} - \boldsymbol{S}$ of the message. Note that this relative address does not depend on the choice of the origin vertex O. This relative

---

[1] An optimal permutation routing on full-duplex hexagonal networks was first given in [40], as a result of a STSM within COST 293.

address is the only information that is added in the heading of the message to be transmitted, constituting in this way the packet to be sent through the network.

Using $i+j+k=0$, the key observation [32] is that if (a,b,c) and (a',b',c') are the relative addresses of two packets, then $(a,b,c) = (a',b',c')$ if and only if there exists a $d \in \mathbb{Z}$ such that $a' = a+d$, $b' = b+d$, and $c' = c+d$.

Definition 10.1. An address $\overrightarrow{SD} = (a,b,c)$ is of the *shortest path form* if there is a path from vertex S to vertex D, consisting of a units of vector i, b units of vector j, and c units of vector k, and this path has the shortest length.

The next result simplifies extraordinarily the routing in hexagonal networks.

Theorem 10.3 ([32]). *An address (a,b,c) is of the shortest path form if and only if at least one component is zero, and any two components do not have the same sign.*

Corollary 10.1 ([32]). *Any address has a unique shortest path form.*

Thus, each address $\overrightarrow{SD}$ written in the shortest path form has at most two nonzero components, and they have different signs. In fact, it is easy to find the shortest path form using the next result.

Theorem 10.4 ([32]). *If $\overrightarrow{SD} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$, then*

$$|\overrightarrow{SD}| = \min(|a-c| + |b-c|, |a-b| + |b-c|, |a-b| + |a-c|).$$

10.2.2 Description of the Algorithm for Hexagonal Networks

Many communication networks are represented by graphs satisfying the following property: for any pair of vertices u and v, the edges of a shortest path from u to v can be partitioned into k disjoint classes according to a well-defined criterion. For instance, we have seen in Section 10.2.1 that on a hexagonal network the edges of a shortest path can be partitioned into *positive* and *negative* ones. Similarly, on a k-circulant graph the edges can be partitioned into k classes according to their length.

In graphs that satisfy this property there exists a natural routing algorithm: route all packets along one class of edges one after another. Surprisingly, in many cases this natural algorithm turns out to be optimal. Optimality for 2-circulant graphs is proved using a static approach in [13], and recently using a dynamic distributed algorithm in [39]. The algorithm is called the *big-foot* algorithm because it routes packets first along *long hops* and then along *short hops* in a 2-circulant graph [13]. There is no restriction on the size of the queues. On the square grid, the *big-foot* idea works as is natural, i.e., an optimal algorithm consists of two phases, moving each packet first horizontally and then vertically.

Based on the observations on the addressing scheme of hexagonal networks, it can be proved that for hexagonal networks this algorithm turns out to be also optimal [40].

The optimal algorithm $\mathscr{A}$ is completely distributed, and can be described as follows. At each node u of the network, perform:

I. Preprocessing phase. If there is a packet at node u, the preprocessing phase consists just of computing the relative address $D - S$ of the message in the shortest path form, and adding this information to the message to complete the packet to be sent. Recall that because of Theorem 10.3, $D - S$ can have no more than one negative component. At each step, when a packet is received at u, its relative address is updated.

II. Transmission phase. Repeat:

   a) If there are packets with negative components, send them immediately along the direction of this component.
   b) If not, for each outgoing edge order the packets according to a decreasing number of remaining steps and send the first packet of each queue.

10.2.3 Correctness, Running Time and Optimality

Algorithm $\mathscr{A}$ is really cheap in terms of computational cost, since the only involved operations are integer addition and comparison among the lengths of the addresses of the packets at each node. Let us briefly discuss correctness and give the main ideas of the proof of optimality.

The rules given by Algorithm $\mathscr{A}$ define two directions of movement for each packet. That is, first of all a packet moves along the direction of its negative component, and then along its positive one. Obviously, if a packet has only positive component, it always moves along this direction. The first key observation is that packets can only wait, possibly, during their last direction. That is because if two packets meet when their first direction is not finished yet, it is easy to check that they must have the same origin node, a contradiction. Thus, in a) there can be at most one packet with a negative component at each outgoing edge; hence, there is no ambiguity. Finally, in b) the packet with maximum remaining length at each outgoing edge is unique, since all these packets are moving along their last direction (their negative component is already finished; otherwise, they would be in a)) and each node is the destination of at most one packet.

Using this algorithm, at every step all packets with maximum remaining distance move, and hence at every step the maximum remaining distance over all packets decreases by 1. Thus, the total running time is at most ℓ_{max}, meeting the lower bound. More details can be found in [40]. In the case of permutation routing, we have proved that the number of steps i at each node is at most ℓ_{max}, but written in this way the algorithm can be applied in a more general routing scenario. In conclusion, the main result can be summarized as follows.

Theorem 10.5 (see [40]). *Algorithm $\mathscr{A}$ is an optimal permutation routing algorithm for full-duplex hexagonal networks.*

Besides minimizing the number of steps, a routing algorithm must also be easy to implement; namely, the routing at each step should be determined efficiently. Recall

that a routing algorithm is called *oblivious* if the path of a packet from v depends only on v and its destination, although the waiting time at an intermediate node may depend on other paths. Furthermore, a *translation invariant* oblivious algorithm is completely determined by paths from the origin.

The obliviousness of $\mathscr{A}$ is straightforward since the routing for each packet depends only on the source and destination nodes. Finally, it is clear that to route a packet only the difference $D - S$ between the source and destination node is needed, and thus we have proved the invariance.

Corollary 10.2 (see [40]). *Algorithm $\mathscr{A}$ is an oblivious, translation invariant, and optimal permutation routing algorithm for full-duplex hexagonal mesh networks.*

10.3 Extensions and Open Problems

The algorithm described in Section 10.2.2 can be extended to find optimal permutation routing algorithms for all the other types of plane grids, as well as for finding near-optimal algorithms for (ℓ, k)-routing for all types of plane grids. The following results can be found in [1]:

1. A tight (also including the constant factor) permutation routing algorithms in full-duplex hexagonal grids and half-duplex triangular and hexagonal grids.
2. A tight (also including the constant factor) r-central routing algorithms in triangular and hexagonal grids. It may be interesting to remark that the optimal algorithms for r-central routing slightly deviate from the general scheme of the permutation routing algorithms.
3. A tight (also including the constant factor) (k, k)-routing algorithms in square, triangular, and hexagonal grids.
4. Approximation algorithms for (ℓ, k)-routing in square, triangular, and hexagonal grids.

Of course, there are many questions to be asked. For instance, the words *approximation algorithms* mentioned above mean that there is no optimal (or even tight) algorithm for (ℓ, k)-routing known for any plane grid. This is definitely the most challenging open problem concerning (ℓ, k)-routing on plane grids.

There are other important avenues for further research. An important issue is to optimize also the maximum size of the queues at the nodes. We have not addressed this point here, mostly because often when optimizing the queue size the running time increases. Finding the right trade-off between both parameters would be a celebrated result.

Finally, observe that all the graphs discussed in Section 10.2.2 are Cayley graphs, with their corresponding generators. In fact, the generators of each type of graph induce the partition of the shortest paths into the k mentioned sets. Therefore, it seems natural that this idea of routing packets one class after another could be extended to any Cayley graph, which would be a quite general result.

References

1. Amini, O., Huc, F., Sau, I., Žerovnik, J.: (ℓ, k)-Routing on Plane Grids. Rapport de Recherche 6480 INRIA (2008)
2. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O.: On-line routing of virtual circuits with applications to load balancing and machine scheduling. Journal of the ACM (JACM) **44**(3), 486–504 (1997)
3. Aspnes, J., Busch, C., Dolev, S., Fatourou, P., Georgiou, C., Shvartsman, A.: Eight Open Problems in Distributed Computing. Bulletin of the EATCS **90**, 109–126 (2006)
4. Awerbuch, B., Azar, Y.: Local optimization of global objectives: competitive distributed deadlock resolution and resource allocation. In: 35th Annual Symposium on Foundations of Computer Science (FOCS), pp. 240–249 (1994)
5. Azar, Y., Cohen, E., Fiat, A., Kaplan, H., Räcke, H.: Optimal oblivious routing in polynomial time. Journal of Computer and System Sciences **69**(3), 383–394 (2004)
6. Borodin, A., Hopcroft, J.: Routing, merging and sorting on parallel models of computation. Proceedings of the fourteenth annual ACM symposium on Theory of computing pp. 338–344 (1982)
7. Busch, C., Magdon-Ismail, M., Xi, J.: Oblivious routing on geometric networks. Proceedings of the 17th annual ACM symposium on Parallelism in algorithms and architectures pp. 316–324 (2005)
8. Busch, C., Magdon-Ismail, M., Xi, J.: Optimal oblivious path selection on the mesh. Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2005), Denver, Colorado, USA, April (2005)
9. Dobravec, T., Robič, B., Žerovnik, J.: Permutation routing in double-loop networks: design and empirical evaluation. Journal of Systems Architecture **48**, 387–402 (2003)
10. Dobravec, T., Žerovnik, J., Robič, B.: An optimal message routing algorithm for circulant networks. Journal of Systems Architecture **52**, 298–306 (2006)
11. Fraigniaud, P., Lazard, E.: Methods and problems of communication in usual networks. Discrete Applied Mathematics **53**, 79–134 (1994)
12. Havill, J. T.: Online Packet Routing on Linear Arrays and Rings. Lecture Notes in Computer Science **2076**, 773–784 (2001)
13. Hwang, F., Lin, T., Jan, R.: A Permutation Routing Algorithm for Double Loop Network. Parallel Processing Letters **7**(3), 259–265 (1997)
14. Hwang, F., Yao, Y., Dasgupta, B.: Some permutation routing algorithms for low-dimensional hypercubes. Theoretical Computer Science **270**, 111–124 (2002)
15. Iwama, K., Kambayashi, Y., Miyano, E.: New Bounds for Oblivious Mesh Routing. Journal of Graph Algorithms and Applications **5**(5), 17–38 (2001)
16. Iwama, K., Miyano, E.: Oblivious Routing Algorithms on the Mesh of Buses. Journal of Parallel and Distributed Computing **60**, 137–149 (2000)
17. Iwama, K., Miyano, E.: An $O(\sqrt{N})$ Oblivious Routing Algorithm for Two-Dimensional Meshes of Constant Queue-Size. Journal of Algorithms **41**, 262–279 (2001)
18. Jan, G. E., Lin, M. B.: Concentration, load balancing, partial permutation routing, and superconcentration on cube-connected cycles parallel computers. Journal of Parallel and Distributed Compututing **65**, 1471–1482 (2005)
19. Kaklamanis, C., Krizanc, D., Tsantilas, T.: Tight bounds for oblivious routing in the hypercube. Theory of Computing Systems **24**(1), 223–232 (1991)
20. Klavžar, S., Vesel, A., Žigert, P.: On resonance graphs of catacondensed hexagonal graphs: structure, coding, and Hamiltonian path algorithm. MATCH Communications in Mathematical and in Computer Chemistry **49**(49), 99–116 (2003)

21. Kranakis, E., Sing, H., Urrutia, J.: Compas Routing in Geometric Graphs. In: 11th Canadian Conference of Computational Geometry, pp. 51–54 (1999)
22. Kunde, M., Niedermeier, R., Rossmanith, P.: Faster sorting and routing on grids with diagonals. In: 11th Symposium of Theoretical Computer Science, 775, pp. 225–236. Lecture Notes on Computer Science (1994)
23. Kunde, M., Tensi, T.: $(k-k)$ Routing on Multidimensional Mesh-Connected Arrays. Journal of Parallel and Distributed Computing **11**, 146–155 (1991)
24. Leighton, F., Maggs, B. M., Richa, A. W.: Fast Algorithms for Finding O(Congestion + Dilation) Packet Routing Schedules. In: 28th Annual Hawaii International Conference on System Sciences, pp. 555–563 (1995)
25. Leighton, F. T., Maggs, B. M., Rao, S. B.: Packet Routing and Job-Shop Scheduling in O(congestion + dilation) Steps. Combinatorica **14**(2), 167–186 (1994)
26. Leighton, T.: Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes. Morgan-Kaufman, San Mateo, California (1992)
27. Leighton, T., Maggs, B., Rao, S.: Universal packet routing algorithms. In: 29th Annual Symposium on Foundations of Computer Science (FOCS), pp. 256–269 (1988)
28. Leighton, T., Makedon, F., Tollis, I. G.: A $2n-2$ Step Algorithm for Routing in an $n \times n$ Array with Constant-Size Queues. Algorithmica **14**, 291–304 (1995)
29. Litman, A., Moran-Schein, S.: Fast, minimal, and oblivious routing algorithms on the mesh with bounded queues. Journal of Interconnection Networks **2**, 445–469 (2001)
30. Maggs, B.: A Survey of Congestion + Dilation Results for Packet Scheduling. 40th Annual Conference on Information Sciences and Systems **22**(24), 1505–1510 (2006)
31. Makedon, F., Symvonis, A.: Optimal algorithms for the many-to-one routing problem on 2-dimensional meshes. Microprocessors and Microsystems **17**, 361–367 (1993)
32. Nocetti, F. G., Stojmenović, I., Zhang, J.: Addressing and Routing in Hexagonal Networks with Applications for Tracking Mobile Users and Connection Rerouting in Cellular Networks. IEEE Transactions on Parallel and Distributed Systems **13**(9), 963–971 (2002)
33. Osterloh, A.: Optimal oblivious routing on d-dimensional Meshes. Theoretical Computer Science **333**, 331–346 (2005)
34. Ostrovsky, R., Rabani, Y.: Universal O(congestion + dilation + $\log^{1+\varepsilon} N$) Local Control Packet Switching Algorithms. In: 29th Annual ACM Symposium on the Theory of Computing, pp. 644–653. New York (1997)
35. Pietracaprina, A., Pucci, G.: Optimal Many-to-One Routing on the Mesh with Constant Queues. Lecture Notes in Computer Science **2150**, 645–649 (2001)
36. Räcke, H.: Exploiting locality for data management in systems of limited bandwidth. In: 38th Annual Symposium on Foundations of Computer Science (FOCS), pp. 284–293 (1997)
37. Räcke, H.: Minimizing congestion in general networks. In: 43rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 43–52 (2002)
38. Rajasekaran, S., Overholt, R.: Constant queue routing on a mesh. Journal of Parallel and Distributed Computing **15**, 160–166 (1992)
39. Robič, B., Žerovnik, J.: Minimum 2-terminal routing in 2-jump circulant graphs. Computers and Artificial Intelligence **19**(1), 37–46 (2000)
40. Sau, I., Žerovnik, J.: An Optimal Permutation Routing Algorithm on Full-Duplex Hexagonal Networks. Discrete Mathematics and Theoretical Computer Science **10**(3), 49–62 (2008)
41. Scheideler, C.: Universal Routing Strategies for Interconnection Networks. Springer (1998)
42. Sibeyn, J.: Routing on Triangles, Tori and Honeycombs. International Journal of Foundations of Computer Science **8**(3), 269–287 (1997)
43. Sibeyn, J., Kaufman, M.: Deterministic 1-k routing on meshes (with applications to wormhole routing). In: LNCS (ed.) 11th Symposium on Theoretical Aspects of Computer Science, vol. 775, pp. 237–248 (1994)
44. Sibeyn, J. F., Chlebus, B. S., Kaufmann, M.: Deterministic Permutation Routing on Meshes. Journal of Algorithms **22**(1), 111–141 (1997)
45. Srinivasan, A., Teo, C. P.: A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria. In: STOC '97: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, pp. 636–643. ACM, New York, NY, USA (1997). DOI http://doi.acm.org/10.1145/258533.258658

46. Stojmenović, I.: Honeycomb Networks: Topological Properties and Communication Algorithms. IEEE Transactions on Parallel and Distributed Systems **8**(10), 1036–1042 (1997)
47. Suel, T.: Routing and Sorting on Meshes with Row and Column Buses. In: Parallel Processing Symposium, vol. Eighth International Volume, pp. 411–417 (1994)
48. Tošić, R., Masulović, D., Stojmenović, I., Brunvoll, J., Cyvin, B., Cyvin, S.: Enumeration of Polyhex Hydrocarbons up to h=17. Journal of Chemical Information and Computer Sciences **35**, 181–187 (1995)
49. Trobec, R.: Two-dimensional regular d-meshes. Parallel Computing **26**, 1945–1953 (2000)
50. Valiant, L., Brebner, G.: Universal schemes for parallel communication. Proceedings of the thirteenth annual ACM symposium on Theory of computing pp. 263–277 (1981)

Part II
Studies in Wireless
and Ad Hoc Networks

Chapter 11
Mathematical Optimization Models for WLAN Planning

Sandro Bosio, Andreas Eisenblätter, Hans-Florian Geerdes, Iana Siomina, and Di Yuan

Abstract Wireless Local Area Networks (WLANs) based on the IEEE 802.11 standard family are used widely for wireless broadband Internet access. The performance aspects of WLANs range from deployment cost, coverage, capacity, interference, and data throughput to efficiency of radio resource utilization. In this chapter, we summarize some recent advances in applying mathematical optimization models for solving planning problems arising in placing access points (APs) and assigning channels in WLANs. For AP location, we present an optimization model aimed at maximizing the average user throughput. For channel assignment, we present two modeling approaches that use different performance metrics. We also discuss integrated models for joint optimization of AP location and channel assignment. We report computational experiments with real-life data, and show the advantages of mathematical optimization in WLAN planning.

Sandro Bosio
Institut für Mathematische Optimierung, Otto-von-Guericke Universität, D-39106 Magdeburg, Germany, e-mail: `bosio@mail.math.uni-magdeburg.de`

Andreas Eisenblätter
atesio GmbH, Sophie-Taeuber-Arp-Weg 27, D-12205 Berlin, Germany, e-mail: `eisenblaetter@atesio.de`, and
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany, e-mail: `eisenblaetter@zib.de`

Hans-Florian Geerdes
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany

Iana Siomina
Ericsson Research, Ericsson AB, Isafjordsgatan 14E, SE-164 80, Stockholm, Sweden, e-mail: `iana.siomina@ericsson.com`

Di Yuan
Department of Science and Technology, Linköping University, SE-601 74, Norrköping, Sweden, e-mail: `diyua@itn.liu.se`

11.1 Introduction

WLAN (Wireless Local Area Network, also WiFi) is a technology for providing wireless broadband Internet access. Due to their low cost and ease of use, WLANs are widely deployed. The network layout and its configuration determine the performance of a WLAN; important performance indicators are deployment cost, coverage, capacity, interference, data throughput, and efficiency of radio resource utilization. At present, WLANs are often installed using rules of thumb; this calls for quantitative and systematic methods that can outperform manual approaches. This chapter presents mathematical optimization approaches to the problems of locating access points (APs) and assigning a channel to each of them. We show how mathematical models can capture the essential features of the technology and provide superior network designs.

Most aspects of WLAN optimization resemble problems studied in mobile cellular network planning. The AP positioning problem is akin to base station positioning and *coverage planning* in cellular networks. Due to the smaller coverage area of WLANs, however, the number of installed APs is smaller, and the signal propagation characteristics are different. Moreover, WLANs are mainly intended for providing broadband access to users that are rather stationary. *Channel assignment* for minimizing interference is needed in WLAN just as, for instance, in GSM (see Section 1.5.5.1). The number of channels is, however, much smaller, and access to the medium is controlled by a contention mechanism on each channel. The impact of this mechanism on network performance is paramount, and requires tailored modeling approaches.

We present several optimization models for the planning phase of deploying WLAN. The models can use both measurements and prediction-based data. As data are collected prior to solving the models, optimization can be performed using centralized computation. We optimize AP location using a facility location model in which the average single-user throughput is maximized. Two modeling approaches are considered for channel assignment: *overlap graphs* and *contention sets*. With overlap graphs, each pair of APs is assigned a weight proportional to the area in which the received signal strengths of the two APs are above some thresholds. Channel assignment has the objective of minimizing the total weighted overlap of APs on the same and adjacent channels. Two integer-linear programming models for minimum-overlap channel assignment are discussed in the chapter. For each user, the contention set comprises all other users potentially contending for medium access with the user. The modeling approach resulting from the concept uses a performance metric, called *network efficiency*, that reflects the average probability of successful access to the medium. Using contention sets, the goal is to maximize the network efficiency metric. Because the contention sets carry information on

medium contention of each individual user, they model WLAN medium access more accurately than AP overlap, which represents medium contention through an aggregated view. On the other hand, maximum-efficiency channel assignment using contention sets leads to complex models. In the chapter, we present a hyperbolic integer programming model for maximum-efficiency channel assignment, and derive a linearization based on the enumeration of contention sets.

In addition to models for performing AP location and channel assignment separately, we present optimization models that integrate AP location with channel assignment. Integrating AP location with minimum-overlap channel assignment leads to an objective function representing a trade-off between overlap minimization and throughput maximization. For the modeling approach using contention sets, the objective of maximizing efficiency can be used for joint AP location and channel assignment.

We report computational experiments for a real-life WLAN planning instance, where the input parameters are based on real throughput measurements and detailed signal-propagation modeling. The input data include a set of potential AP locations in an office building, a set of available channels, a set of test points representing expected user locations, and signal propagation data between candidate AP locations and test points. The strengths and the capabilities of both the sequential approaches and of the integrated model are assessed. In our experiments, the WLAN designs obtained from the optimization models considerably outperform the manual planning solution which has been deployed in the network.

The remainder of the chapter is organized as follows. Some technical background of WLANs is provided in Section 11.2; the related work on WLAN planning is reviewed in Section 11.3. Mathematical notation and definitions used in the chapter are introduced in Section 11.4. Section 11.5 discusses the optimization of AP locations. The models for minimum-overlap channel assignment are presented in Section 11.6. The concept of contention set and the resulting models are presented in Section 11.7. The integration of AP location and channel assignment is then discussed in Section 11.8. Computational experiments are presented in Section 11.9. Finally, Section 11.10 gives some conclusions and an outlook on future developments.

11.2 Technical Background

The WLAN technology is defined in the IEEE 802.11 standard [16], specifying a number of possible physical-layer implementations as well as many networking aspects. At present, WLANs using IEEE 802.11b and the more recent IEEE 802.11g are the most popular. (These former amendments have now been joined in the latest standard.) They use different physical-layer implementations on the same frequency band at 2.4 GHz. IEEE 802.11g is backward-compatible, so that hardware implementing the two standards can interoperate.

11.2.1 Physical Layer

WLANs employ *rate adaptation* to utilize the radio channel efficiently. According
to the received signal strength, the raw data rate on a link is selected from a discrete
set ranging up to 54 Mbit/s (11 Mbit/s for IEEE 802.11b). Due to protocol overhead,
however, the net data rate is much lower (see Figure 11.4).

Both the IEEE 802.11b and 802.11g standards divide the 2.4 GHz spectrum
into 13 channels.[1] Two neighboring channels are separated by 5 MHz, as shown
in Figure 11.1. Channels with at least 24 MHz separation are considered to be
non-overlapping. Otherwise, they are called *adjacent* or *overlapping* channels. The
spectrum available to WLAN provides at most three non-overlapping channels (for
example, channels 1, 6, and 11; see Figure 11.1). Since the 2.4 GHz spectrum is
unlicensed, a WLAN may also be subject to the interference of external radiation
sources (e.g., other WLANs, microwave ovens, cordless telephones).
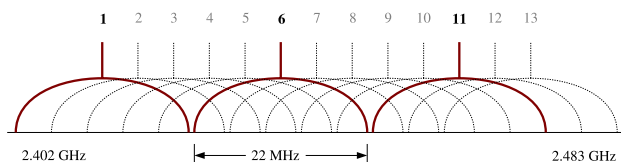
Fig. 11.1 Channels specified by the IEEE 802.11b/g standards

11.2.2 Architecture

A *station* is a device containing an IEEE 802.11 network interface card (NIC). A
WLAN connects stations either in *ad hoc* mode or in *infrastructure* mode. A *basic
service set* (BSS) is a set of stations that can communicate either directly or through
some other station. If stations are able to communicate directly, we speak of an *in-
dependent basic service set* (IBSS). If a self-contained network is thus formed, this
is referred to the *ad hoc mode* of WLAN. In graph terminology, the stations of a
WLAN in ad hoc mode form a clique, assuming that no station has been config-
ured to perform network layer routing for other stations. In the *infrastructure mode*,
the stations in a BSS communicate through an *access point* (AP), which is itself a
station. In the rest of this section, we will refer to user stations simply as *users*, or
mobile terminals (MTs).

---

[1] Channel availability varies by country due to radio spectrum regulation. The 13 channels are
typically available in the European Telecommunications Standards Institute (ETSI) regulatory do-
main. One additional IEEE 802.11b channel is available in Japan. In North America, the Federal
Communications Commission (FCC) and Industry Canada (IC) restrict the spectrum usage to 11
of the 13 channels.

In the infrastructure mode, several BSSs are typically connected to a wired Ethernet backbone, called the *distribution system* (DS). The DS enables mobility support and seamless integration of multiple BSSs. BSSs connected via the DS form an *extended service set* (ESS) of a WLAN; see Figure 11.2. ESSs are used for broadband Internet access, and are the focus of this chapter. Every user accesses the DS through one AP of the ESS, with which the user is said to be *associated*. To announce its presence, an AP periodically broadcasts special messages, called *beacon* frames, on the frequency channel used by the AP. A user station seeks beacon frames on all WLAN channels, and selects one AP for association. The 802.11 standard does not specify an algorithm for selecting which of the available APs to associate with. Typically, the station chooses the AP from which the beacon frame is received with the best signal strength. By this choice, the station maximizes the expected data rate (see Section 11.4). Some research has been conducted on alternative variants [3, 24]. The frequency channel used by the AP is inherited by all its associated users.
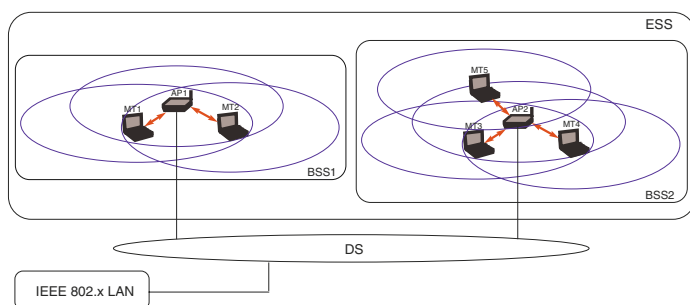
Fig. 11.2 Infrastructure mode of WLAN.

11.2.3 Medium Access Control

Unlike cellular networks (e.g., GSM or UMTS) where users obtain a dedicated resource in form of frequency, time slot, or channelization code, WLAN applies randomized medium access, and uses a *medium access control* (MAC) mechanism to deal with collisions and medium contention. The MAC mechanism is used in both downlink (AP to user station) and uplink (user station to AP) directions. A *collision* occurs when the signals of two or more transmitting stations overlay at either of the receiving stations. *Medium contention* occurs when two (or more) stations compete for using the medium. Stations transmitting on non-overlapping channels will not cause collision or medium contention.

The IEEE 802.11 protocol defines two MAC mechanisms: the Distributed Coordination Function (DCF) and the Point Coordination Function (PCF). PCF is centralized and can therefore, in theory, avoid collision and medium contention at the

cost of coordination overhead. Implementing PCF is, however, optional, and DCF is wider spread, so we focus on DCF. In DCF, stations negotiate medium access among themselves in a distributed and randomized access scheme using a *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA) protocol.

In CSMA, before initiating a transmission, a station first *senses* the medium. If a signal from another station is detected on the same frequency channel, the station waits for the ongoing transmission to finish before transmitting. This mechanism avoids some collisions, but unresolved cases remain: In the *hidden terminal* scenario, illustrated in Figure 11.3(a), a terminal (MT1 in the figure) senses the carrier idle and starts transmission concurrently with another one (MT2). Neither of them will detect the other's signal because they are outside the respective radio ranges. The result is a collision at the AP. Another interesting interference scenario is the *exposed terminal* scenario, represented in Figure 11.3(b): A user MT2 needs to transmit to its associated access point AP2, but a second access point AP1, having MT2 within the radio range, is transmitting to some other user MT1. Even though the two transmissions could in fact take place simultaneously without collision (as MT2 is not within radio range of MT1, and AP2 is not within radio range of AP1), MT2 is not allowed to start transmitting, because it is exposed to an ongoing transmission.

(a) Hidden terminal problem   (b) Exposed terminal problem
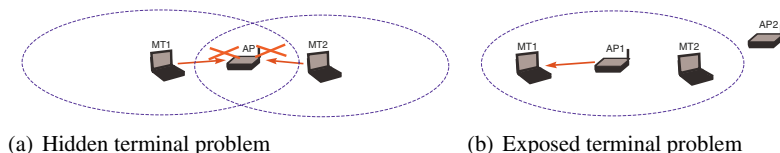
Fig. 11.3 The hidden terminal problem and the exposed terminal problem. Radio ranges are indicated by dotted ovals

CA is used to further reduce the probability of collisions. A receiving station acknowledges each successful transmission. The absence of acknowledgment triggers a retransmission. To reduce the probability of repeated collision, CSMA/CA uses an exponential backoff scheme: If a station wishing to transmit senses the channel idle, it senses for an additional, short period of time (the *Distributed Inter-Frame Space*, DIFS). If the channel gets busy during the DIFS, the station keeps sensing the channel until it is idle for a DIFS, and then generates a random backoff interval from a range called the *contention window*. The contention window size is doubled after each unsuccessful attempt and reset after a successful transmission. The backoff counter is decremented as long as the channel is sensed idle, frozen when the channel is busy, and reactivated (without resetting) when the channel is again idle for more than a DIFS. The transmission starts when the backoff counter reaches zero. In addition to the two-way handshaking (transmission and acknowledgment) technique, DCF defines an optional four-way handshaking technique known as CSMA/CA with Request to Send (RTS) and Clear to Send (CTS). RTS and CTS are short packets used to reserve the channel prior to payload trans-

mission. The RTS/CTS packet exchange, referred to as *virtual carrier sensing,* is advantageous because collisions virtually occur only for these small signaling packets, and the exposed terminal problem and the hidden terminal problem are resolved (if all stations have the same radio range).

11.2.4 Planning Tasks and Performance Aspects

The main decisions in WLAN planning are *AP location* and *channel assignment.* These decisions should be taken with the objective of obtaining good *network performances*. We shall now elaborate on these points.

Determining AP locations is usually the first step in a WLAN deployment process. Locations should be chosen so that the resulting WLAN provides coverage to the intended service area. This requires signal strength measurements and/or predictions of radio propagation. Coverage planning for large buildings with multiple floors is more involved: An AP located on one floor of the building may provide signal coverage, but sometimes also interference, to adjacent floors of the same building, or even to other buildings [14, 15]. Potential interference from neighboring WLANs has to be considered as well.

Distributed and online channel assignments are presently not implemented in APs. A WLAN typically uses a static channel assignment, or each AP uses a simple heuristic to select a channel when powered on. Typically, the heuristic searches for the least congested channel or the channel with the least amount of interference. Using the three non-overlapping channels 1, 6, and 11 is encouraged [8], since stations transmitting on non-overlapping channels do not cause collision or contend for the medium. In this chapter we focus on the case where a static channel assignment is determined during the planning.

From a user standpoint, the most important WLAN performance indicator is the data throughput. Since the bandwidth of the DS is typically much higher than that of the radio interface, the radio link is the performance-limiting point. The data throughput on the radio link depends on the net throughput experienced by the user when holding the medium (which we call *single-user throughput*), on the number of users associated with the same AP and their activity, and on interference due to users associated with other APs. The single-user throughput is essentially determined by the distance (attenuation) to the associated AP, and is easy to model. An appropriate modeling of interference and contention, on the other hand, is the main task in designing optimization approaches to WLAN planning. In this chapter, the model for optimizing AP location (Section 11.5) focuses on the aspect of single-user throughput. The model can be applied to both downlink and uplink directions. For channel assignment, the first modeling approach (Section 11.6) uses an overlap graph to provide an aggregated view of interference and contention. This approach does not require us to distinguish between downlink and uplink. The second approach (Section 11.7) models in detail medium contention for every individual user, and the resulting model is suitable for addressing performance of the uplink.

11.3 Related Work

AP location has been occasionally studied separately [21, 32, 36], with the objective of optimizing signal coverage and quality, and without referring to the CSMA/CA mechanism. The WLAN channel assignment problem is related to GSM frequency assignment problems [1, 2, 23], notably to the minimum-interference frequency assignment (see Section 1.5.5.1). For GSM, graph-coloring techniques are used [12]; the graph-coloring approach is applied to WLAN in static [29] and dynamic [31, 37] settings. Some contributions [26] use a more accurate model of MAC level performance for channel assignment. Some NP-hardness results are known [26, 29] for the channel assignment problem.

In conjunction, channel assignment and AP location for WLAN have often been treated sequentially [14, 39], in a multi-objective setting [17]. Search heuristics are most popular for these problems. Custom algorithms have been developed in [27, 30]. Integer programming methods are also used [25, 28, 40]. The channel assignment is, however, typically treated as a feasibility problem constraining an essentially coverage-driven approach. In this chapter, Sections 11.5, 11.6, and 11.8.1 are based on [11, 33, 34], while Sections 11.7 and 11.8.2 are based on previous work on efficiency optimization in single- and multiple-frequency WLANs [4–6].

11.4 Notation and Definitions

Let us introduce some mathematical notation to be used in our models. The set of candidate AP locations is denoted by $\mathscr{A} = \{1,\ldots,A\}$, and we denote by $\mathscr{A}^2 = \{(a,b) : a,b \in \mathscr{A}, a < b\}$ the set of ordered pairs of APs in $\mathscr{A}$. Given two distinct APs a,b, exactly one of (a,b) and (b,a) is in $\mathscr{A}^2$. The maximum number of installed APs is denoted by M. The service area is represented by a set of test points (TPs) $\mathscr{I} = \{1,\ldots,I\}$, where each element $i \in \mathscr{I}$ represents a potential user location. In the remainder, the size of an area is measured by the number of TPs it contains.

For each AP $a \in \mathscr{A}$ we define a serving range, so that TPs within the serving range of an AP can be served by the AP. For every TP $i \in \mathscr{I}$, we use $\mathscr{A}_i$ to denote the set of APs for which i is within the serving ranges. Representing the same information from the AP point of view, we denote by $\mathscr{I}_a$ the set of TPs that can be served (or covered) by AP $a \in \mathscr{A}$.

For every station (AP or TP) we define a second signal range, commonly referred to as carrier sense range. A station is within carrier sensing range from some other station if they operate on the same frequency channel and if the former, performing carrier sensing while the latter is transmitting, senses the channel as idle. This implies that stations within each other's carrier sense range have to contend for the medium. For APs, the carrier sense range is at least as large as the serving range.

As discussed in Section 11.2.4, one important WLAN performance metric is the single-user throughput, defined as the maximum achievable throughput when

a user communicates with its associated AP without contention. We denote by t_{ai} the single-user throughput of TP i when it is associated with AP a.

The set of available channels is denoted by $\mathscr{C} \subseteq \{1, \ldots, 13\}$, and the distance (in MHz) between two channels $c_1, c_2 \in \mathscr{C}$ by $|c_1 - c_2|$. We denote by $\mathscr{D} = \{|c_1 - c_2| : c_1, c_2 \in \mathscr{C}, |c_1 - c_2| < 24 \text{ MHz}\}$ the set of channel distances between overlapping channels. For example, if only the non-overlapping channels are allowed, $\mathscr{C} = \{1, 6, 11\}$ and $\mathscr{D} = \{0\}$, while if all the channels can be used then $\mathscr{C} = \{1, \ldots, 13\}$ and $\mathscr{D} = \{0, 5, 10, 15, 20\}$ (due to the 5 MHz channel separation). For the sake of simplicity, we assume that $\mathscr{C}$ does not vary by AP.

When channel assignment is considered, the set of installed APs is denoted by $\bar{\mathscr{A}} \subseteq \mathscr{A}$, and the set of its pairs by $\bar{\mathscr{A}}^2 \subseteq \mathscr{A}^2$. We assume that each TP $i \in \mathscr{I}$ associates with the AP providing the highest single-user throughput $t_i = \max_{a \in \bar{\mathscr{A}}}\{t_{ai}\}$, and we denote such an AP by $\bar{a}(i)$.

At the physical layer, the value of t_{ai} (and t_i) is dependent on the received signal strength. At the radio interface, this throughput corresponds to one of the nominal data rates defined in IEEE 802.11 standards. From an application standpoint, the throughput should be defined as the end-to-end data throughput, for which unreliable radio propagation, protocol behavior, and overhead at the link layer and above are the constraining factors. In either case, the throughput as a function of received signal strength can be found using a predication model or experimentally. We illustrate the latter for IEEE 802.11g using an AP of type Cisco AP-1200/AP21G [7] and the network benchmarking tool NETIO. The transport layer protocol is the Transmission Control Protocol (TCP). The TCP end-to-end throughput is measured by continuously transmitting segments of 1 kB in size in one minute between a laptop and a PC connected to the DS. The results are shown in Figure 11.4. The figure shows both the physical-layer data rates on the radio interface as well as the TCP end-to-end throughput. Experiments also show that the results are valid for both downlink (AP to laptop) and uplink (laptop to AP). The measurement points of the end-to-end throughput can be interpolated by means of a best-fit polynomial (illustrated by the dotted curve in Figure 11.4), which we use to define t_{ai} for the instance described in Section 11.9.
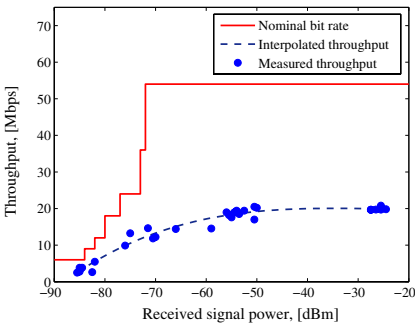
Fig. 11.4 Nominal bit rate and end-to-end throughput as functions of received signal strength

11.5 AP Location Optimization

The AP location problem amounts to selecting a subset $\bar{\mathscr{A}}$ of the candidate locations $\mathscr{A}$ for installing APs. It can be modeled as a variation of the facility location problem. A natural objective function is to maximize the total single-user throughput over all TPs. We introduce two sets of variables:

$$
\begin{aligned}
z_a &= \begin{cases} 1 & \text{if AP } a \text{ is installed,} \\ 0 & \text{otherwise.} \end{cases} \\
x_{ai} &= \begin{cases} 1 & \text{if TP } i \text{ is served by AP } a, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
\tag{11.1}
$$

The model of maximum-throughput AP location (**MTAL**) reads

$$
\max \quad \frac{1}{I} \sum_{i \in \mathscr{I}} \sum_{a \in \mathscr{A}_i} t_{ai} x_{ai} \tag{11.2a}
$$

$$
\text{s.t.} \quad x_{ai} - z_a \le 0 \qquad\qquad i \in \mathscr{I}, a \in \mathscr{A}_i \tag{11.2b}
$$

$$
\sum_{a \in \mathscr{A}_i} x_{ai} \le 1 \qquad\qquad i \in \mathscr{I} \tag{11.2c}
$$

$$
\sum_{a \in \mathscr{A}} z_a \le M \tag{11.2d}
$$

$$
z_a \in \{0,1\} \qquad\qquad a \in \mathscr{A} \tag{11.2e}
$$

$$
x_{ai} \in \{0,1\} \qquad\qquad i \in \mathscr{I}, a \in \mathscr{A}_i \tag{11.2f}
$$

In MTAL, the objective function (11.2a) is to maximize the average throughput (and equivalently the total throughput) of the TPs. Constraints (11.2b) ensure that a TP can be served by an AP only if the latter is installed. By constraints (11.2c), a TP can be associated with and served by at most one AP. The next constraint sets a maximum limit on the number of installed APs.

Note that there is no constraint requiring that all TPs have to be covered. Rather, the model encourages coverage by rewarding a higher throughput. Furthermore, in any optimal solution each TP will always be served by the AP providing the best signal strength and equivalently the highest throughput value. Observe also that there is always an optimal solution in which constraint (11.2d) is tight, as adding APs to a solution does not reduce the throughput. Finally, although the combinatorial problem represented by MTAL is generally NP-hard, in most of the practical cases of WLAN planning the number of candidate AP locations is quite small (often less than a hundred), and MTAL can be solved by commercial integer-linear solvers in a short time.

11.6 Minimum-Overlap Channel Assignment

Given a set $\bar{\mathscr{A}} \subseteq \mathscr{A}$ of installed APs, the next step is to assign one of the available channels to each AP. With 13 available channels as shown in Figure 11.1, two basic selections of the set of candidate channels are the complete set $\mathscr{C} = \{1, \ldots, 13\}$, and the set of non-overlapping channels $\mathscr{C} = \{1, 6, 11\}$ for which only co-channel overlap may occur. We will first present a channel assignment model for the general case, and then a model specific to the case of three non-overlapping channels.

In this section, we treat channel assignment in a similar way as frequency assignment in cellular networks. Instead of trying to model CSMA/CA explicitly, we model interference and medium contention indirectly by considering the amount of overlap between APs operating on the same channel or adjacent channels. To do so, we define a parameter v_{ab} to represent the number of TPs at which the received signals of APs a and b are both detectable. More precisely, v_{ab} is the number of TPs that lie within the carrier sense ranges of both APs and in the serving range of at least one of them. We call v_{ab} the *co-channel overlap* if APs a and b are assigned the same channel. If the two APs operate on adjacent channels at a distance d, the parameter is scaled by a weight $F(d) \in [0, 1]$. In our computational experiments we use $F(d) = 1/(1 + d/5 \text{ MHz})^k$ if $d \leq 24$ MHz, and 0 otherwise. Note that $F(0) = 1$ and that $F(0)$ models the impact of contention and co-channel interference on the system performance, while $F(d > 0)$ addresses the inter-channel interference issue, which is less critical but may still have a significant effect on the performance when channel distance is small. The parameter k can be used to model the impact of overlap due to adjacent channels; in our experiments we set $k = 2$. For an alternative definition of $F(d)$, derived empirically, see [29, 38]. Also, instead of using the size of overlap between AP pairs, another modeling approach is to consider for each TP, the covering APs and their signal strengths. For channel assignment in GSM, such an approach (e.g., [18]) amounts to aggregating multiple interfering sources.

To formulate the problem, we introduce the following variables:

$$
\begin{aligned}
f_a^c &= \begin{cases} 1 & \text{if AP } a \text{ operates on channel } c, \\ 0 & \text{otherwise.} \end{cases} \\
w_{ab}^d &= \begin{cases} 1 & \text{if the channel distance between AP } a \text{ and AP } b \text{ equals } d, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
\tag{11.3}
$$

Variables w_{ab}^d translate channel assignment into overlap. The minimum weighted-overlap channel assignment problem (MOCA) can then be modeled as follows.

$$\min \quad \sum_{(a,b)\in\mathscr{A}^2}\sum_{d\in\mathscr{D}} v_{ab}F(d)\,w_{ab}^d \tag{11.4a}$$

$$\text{s.t.} \quad \sum_{c\in\mathscr{C}} f_a^c = 1 \qquad\qquad\qquad a\in\bar{\mathscr{A}} \tag{11.4b}$$

$$w_{ab}^{|c_1-c_2|} \geq f_a^{c_1} + f_b^{c_2} - 1 \quad (a,b)\in\mathscr{A}^2, c_1,c_2\in\mathscr{C}, |c_1-c_2|\in\mathscr{D} \tag{11.4c}$$

$$w_{ab}^d \in \{0,1\} \qquad\qquad (a,b)\in\mathscr{A}^2, d\in\mathscr{D} \tag{11.4d}$$

$$f_a^c \in \{0,1\} \qquad\qquad a\in\bar{\mathscr{A}}, c\in\mathscr{C} \tag{11.4e}$$

Constraints (11.4b) ensure that one channel is chosen for each AP, and constraints (11.4c) relate channel assignment to the overlap variables. Recall that $\mathscr{D}$ contains distances between overlapping channels only, as we do not need to consider distances d for which $F(d) = 0$. The objective function (11.4a) is the total weighted overlap over all pairs of APs. Note that the problem represented by MOCA is a type of minimum interference frequency assignment problem that has been studied, for example, in [1, 10, 12, 23].

If the candidate channels are non-overlapping, i.e., if $\mathscr{D} = \{0\}$, the problem can be alternatively formulated as a *minimum K-partition* problem, with $K = 3$. Only variables w_{ab}^0 are used, and we drop for simplicity the index 0. The resulting formulation, denoted by 3-MOCA, is the following.

$$\min \quad \sum_{(a,b)\in\mathscr{A}^2} v_{ab}w_{ab} \tag{11.5a}$$

$$\text{s.t.} \quad \sum_{a,b\in Q\,:\,(a,b)\in\mathscr{A}^2} w_{ab} \geq 1 \qquad Q\subseteq\bar{\mathscr{A}}: |Q| = 4 \tag{11.5b}$$

$$w_{ac} \geq w_{ab} + w_{bc} - 1 \qquad (a,b),(b,c)\in\mathscr{A}^2 \tag{11.5c}$$

$$w_{bc} \geq w_{ab} + w_{ac} - 1 \qquad (a,b),(b,c)\in\mathscr{A}^2 \tag{11.5d}$$

$$w_{ab} \geq w_{ac} + w_{bc} - 1 \qquad (a,b),(b,c)\in\mathscr{A}^2 \tag{11.5e}$$

$$w_{ab} \in \{0,1\} \qquad\qquad (a,b)\in\mathscr{A}^2 \tag{11.5f}$$

The objective function (11.5a) of 3-MOCA minimizes the total co-channel overlap. By (11.5b), for any group of four APs, at least two have to use the same channel. Constraints (11.5c)–(11.5e) are referred to as the triangle inequalities. They ensure the transitivity of channel assignment: For any group of three APs, if any two pairs of them use the same channel, then the same is true for the third pair. To translate a feasible solution of 3-MOCA to a channel assignment, let the APs be represented by nodes of a graph. The graph's edges correspond to the nonzero w-variables in the solution. It is easy to see that the graph has at most three disconnected components, each being a clique. The APs of every clique then operate on the same channel.

Model 3-MOCA eliminates the symmetry of MOCA (formed by the f-variables). On the other hand, the underlying idea of 3-MOCA results in model size that grows exponentially in the number of channels [10]. For three non-overlapping channels, however, finding the optimum to 3-MOCA is typically computationally feasible.

11.7 Maximum-Efficiency Channel Assignment

The AP overlap view models the CSMA/CA protocol behavior coarsely, because it aggregates the interference and contention between users into the amount of overlap between APs. Hence, the contention experienced by every individual user is not modeled explicitly. A more detailed analysis is to consider the contention sets. The contention set of a TP is the set of test points potentially contending for medium access with it; this set has a pivotal influence on user throughput achieved under CSMA/CA. From the discussion in Section 11.2, two TPs will contend for the medium if they use the same channel, or more precisely, if the APs with which they are associated use the same channel, and if at least one of the following conditions holds:

1. The two TPs are associated with the same AP.
2. The two TPs are located within the carrier sense range of each other, and hence one will sense the carrier as busy when the other is transmitting.
3. One of the two TPs is within the carrier sense range of the AP with which the other TP is associated; this corresponds to the exposed terminal problem.

If the two TPs use adjacent channels at some distance $d > 0$, then $F(d)$ can be considered as the likelihood that interference will occur (recall that $F(0) = 1$, and $F(d) = 0$ for non-overlapping channels). We denote by $d(i,h)$ the channel distance between the TPs i and h (i.e., between APs $\bar{a}(i)$ and $\bar{a}(h)$).

For each TP $i \in \mathscr{I}$, let $\mathscr{F}_i = \{h \in \mathscr{I} \setminus \{i\} : \bar{a}(h) = \bar{a}(i)\}$ be the set of TPs which contend for the medium with i due to condition 1, and let $\mathscr{P}_i$ be the set of all other potential contending TPs. The set $\mathscr{P}_i$ is derived from the input data by assuming that all APs operate on one single channel, and taking TPs $h \notin \mathscr{F}_i$ for which at least one of conditions 2 and 3 holds. Based on these contention sets, we define an efficiency metric for every TP. Since a user can access the network only if no other user is contending for the medium, the efficiency metric of a TP $i \in \mathscr{I}$ is defined as the throughput t_i scaled by the probability of successfully accessing the medium, which is defined as

$$\frac{1}{1 + |\mathscr{F}_i| + \sum_{h \in \mathscr{P}_i} F(d(i,h))}.$$

The overall network efficiency is the sum of the efficiency values over all TPs. This leads to a maximum-efficiency channel assignment problem. Note that the contention relation is symmetric for every pair of TPs. Moreover, if we consider only non-overlapping channels, for which F gives binary values, the denominator is the cardinality of the contending set plus the TP itself.

11.7.1 A Hyperbolic Model and Linear Reformulations

Consider variables f_a^c and w_{ab}^d defined for MOCA. For two TPs $i, h \in \mathscr{I}$, let us denote by ω_{ih}^d and ω_{hi}^d the variable w_{ab}^d, with $a = \min\{\bar{a}(i), \bar{a}(h)\}$ and $b = \max\{\bar{a}(i), \bar{a}(h)\}$. A model for the problem of finding a channel assignment that maximizes the average efficiency, denoted by MECA, reads:

$$\max \quad \frac{1}{I} \sum_{i \in \mathscr{I}} \frac{t_i}{1 + |\mathscr{F}_i| + \sum_{h \in \mathscr{P}_i} \sum_{d \in \mathscr{D}} F(d) \omega_{ih}^d}$$

$$\text{s.t.} \quad (11.4\text{b}), (11.4\text{c}), (11.4\text{d}), (11.4\text{e})$$

The hyperbolic objective function (sum of ratios) in MECA is the average efficiency over all TPs. As for MOCA, for three non-overlapping channels we have the alternative model 3-MECA, where $w_{ab} = w_{ab}^0$ and $\omega_{ih} = \omega_{ih}^0$.

$$\max \quad \frac{1}{I} \sum_{i \in \mathscr{I}} \frac{t_i}{1 + |\mathscr{F}_i| + \sum_{h \in \mathscr{P}_i} \omega_{ih}}$$

$$\text{s.t.} \quad (11.5\text{b}), (11.5\text{c}), (11.5\text{d}), (11.5\text{e})$$

Problem MECA is NP-hard, even when restricted to the special case 3-MECA. This can be shown by a polynomial-time reduction from 3-coloring. Given an undirected graph $G = (V, E)$, for each node $v \in V$ we create an AP location a_v and a TP i_v, and define the covering set of each AP a_v as $\mathscr{I}_{a_v} = \{i_u : \{v, u\} \in E\} \cup \{i_v\}$. The rates are then defined so that $t_{a_v i_v} = 1$ for every $v \in V$ and $t_{a_u i_v} = \varepsilon$ for every $\{u, v\} \in E$, with $\varepsilon < 1$. Hence $\bar{a}(i_v) = a_v$ and $t_i = 1$. It is then easy to see that G admits a 3-coloring if and only if MECA admits a frequency assignment with objective function value 1.

Solving the hyperbolic formulations MECA and 3-MECA is very difficult. As these are constrained problems, solution approaches for unconstrained hyperbolic problems (e.g., [13]) cannot be applied. It is thus justifiable to look for a linear reformulation. Consider problem 3-MECA. A standard way of linearizing a hyperbolic objective function is to introduce a new variable for each fraction. In our case this consists of adding a continuous variable s_i for each TP $i \in \mathscr{I}$, thus redefining the objective function as $\sum_{i \in \mathscr{I}} t_i s_i$. The resulting hyperbolic constraint $s_i = 1/(1 + |\mathscr{F}_i| + \sum_{h \in \mathscr{P}_i} \omega_{ih})$ can be linearized in two ways. The first linearization is to introduce, for each TP $i \in \mathscr{I}$, the quadratic constraint $1 = (1 + |\mathscr{F}_i|)s_i + \sum_{h \in \mathscr{P}_i} \omega_{ih} s_i$, and linearize the bilinear product $\omega_{ih} s_i$ by standard techniques. The second linearization consists of introducing, for each TP $i \in \mathscr{I}$ and for each possible value $1/(1 + |\mathscr{F}_i| + m)$ of s_i, a binary variable r_{im}, for $m \in \{0 \ldots |\mathscr{P}_i|\}$, and including into the formulation the following constraints:

$$s_i = \sum_{m=0}^{|\mathscr{P}_i|} \frac{r_{im}}{1+|\mathscr{F}_i|+m} \qquad\qquad i \in \mathscr{I}$$

$$\sum_{m=0}^{|\mathscr{P}_i|} r_{im} = 1 \qquad\qquad i \in \mathscr{I}$$

$$\sum_{m=0}^{|\mathscr{P}_i|} m \cdot r_{im} = \sum_{h \in \mathscr{P}_i} \omega_{ih} \qquad\qquad i \in \mathscr{I},$$

which guarantee that if TP i has m contending TPs among those in $\mathscr{P}_i$, then $r_{im} = 1$.

The linear programming (LP) relaxation of the first linearization can be improved by exploiting a disjunctive argument. The LP relaxation of the second linearization can be improved by removing variables r_{im} for which the value $1/(1+m)$ cannot be attained by s_i in any feasible solution. Even after the tightening, however, both the LP relaxations are outperformed by that of an integer-linear model derived from an enumeration of medium contention scenarios, which we present in the next section.

11.7.2 An Enumerative Integer Linear Model

Let us consider problem 3-MECA. A medium contention scenario for a TP $i \in \mathscr{I}$ is defined by the set $\mathscr{F}_i$ and a subset of the TPs in $\mathscr{P}_i$ that use the same frequency channel as i. The idea is to construct, for every TP, the set of all the possible medium contention scenarios, i.e., the set of all the subsets of $\mathscr{P}_i$. But since there could be TPs in $\mathscr{P}_i$ associated with the same AP as i, not all the subsets of $\mathscr{P}_i$ represent possible contention scenarios. It is therefore more efficient to enumerate the subsets of the set $\mathscr{B}_i = \{a \in \bar{\mathscr{A}} : a = \bar{a}(h) \text{ for some } h \in \mathscr{P}_i\}$ with which TPs in $\mathscr{P}_i$ may be associated. Since $\mathscr{P}_i$ is defined on the carrier sense range of TP i, the set $\mathscr{B}_i$ may have some APs that are not included in $\bar{\mathscr{A}}_i$.

Let $\mathscr{S}_i = \{S : S \subseteq \mathscr{B}_i\}$ be the set of all contention scenarios for TP i. Note that for every contention scenario $S \in \mathscr{S}_i$, the corresponding objective function value for TP i, denoted by d_{iS}, is known. To represent the resulting medium contention of a channel assignment, the following variables are introduced.

$$v_{iS} = \begin{cases} 1 & \text{if the channel assignment results in scenario } S \in \mathscr{S}_i \text{ for TP } i, \\ 0 & \text{otherwise.} \end{cases}$$

The enumerative linearization, denoted by 3-MECA$_E$, is the following:

$$\max \quad \frac{1}{I} \sum_{i \in \mathscr{I}} \sum_{S \in \mathscr{S}_i} d_{is} v_{is}$$

$$\text{s.t.} \quad (11.5b), (11.5c), (11.5d), (11.5e)$$

$$\sum_{S \in \mathscr{S}_i} v_{is} = 1 \qquad\qquad i \in \mathscr{I} \qquad\qquad (11.8a)$$

$$w_{\bar{a}(i)b} = \sum_{S \in \mathscr{S}_i : b \in S} v_{is} \qquad\qquad i \in \mathscr{I}, b \in \mathscr{B}_i \qquad\qquad (11.8b)$$

$$v_{is} \in \{0, 1\} \qquad\qquad i \in \mathscr{I}, S \in \mathscr{S}_i$$

Constraints (11.8a) state that exactly one scenario must be chosen for every TP, while constraints (11.8b) make sure that for every $b \in \mathscr{B}_i$ the variable $w_{\bar{a}(i)b}$ has value 1 if and only if the scenario selected for TP i includes AP b. This imposes coherence among w and v_{is}. A similar model can be derived for problem MECA. In this case, a medium contention scenario is represented by a partition of $\mathscr{B}_i$ into $|\mathscr{D}|$ subsets, one for each channel distance.

An obvious difficulty in the model 3-MECA$_E$ is that the number of v-variables grows exponentially fast in the size of $\mathscr{B}_i$. However, $\mathscr{B}_i$ is typically small in size when planning channel assignment, since only a small number of APs are installed out of all the possible candidate locations. Moreover, since in an optimal solution typically only few overlapping APs share the same frequency, it is possible to create a restricted formulation by considering only elements $S \in \mathscr{S}_i$ having cardinality less than or equal to a given parameter. An alternative approach is to apply column generation to the LP relaxation of 3-MECA$_E$, and to use branch-and-price to search for an optimal solution.

11.8 Integrated Planning of AP Location and Channel Assignment

We now introduce integrated optimization models that decide both AP locations and channel assignments simultaneously.

11.8.1 AP Location and Minimum-Overlap Channel Assignment

Throughput and overlap have been used as objectives in the two tasks, and both objectives obviously antagonize each other. This calls for multi-criteria optimization techniques. We use the popular weighted-sum scaling technique [9]: A trade-off parameter $\alpha \in (0, 1)$ is introduced to weigh the two objectives for AP location and channel assignment. Because the two objectives are of different dimension, we introduce an additional scaling parameter K whose value is found empirically. The integration of MTAL and MOCA (with $\bar{\mathscr{A}} = \mathscr{A}$) results in the following model,

denoted by MT-MO:

$$\max \quad (1-\alpha)K \sum_{i\in\mathscr{I}} \sum_{a\in\mathscr{A}_i} t_{ai}x_{ai} - \alpha \sum_{(a,b)\in\mathscr{A}} \sum_{d\in\mathscr{D}} v_{ab}F(d)\, w_{ab}^d$$

$$\text{s.t.} \quad \sum_{c\in\mathscr{C}} f_a^c = z_a \qquad\qquad a\in\mathscr{A} \qquad\qquad (11.9)$$

$$(11.2\text{b}),(11.2\text{c}),(11.2\text{d}),(11.2\text{e}),(11.2\text{f})$$

$$(11.4\text{c}),(11.4\text{d}),(11.4\text{e}).$$

In MT-MO, AP location and channel assignment interact in the objective function and in (11.9); the latter states that a channel needs to be assigned to an AP if and only if the AP is installed.

Another integrated model can be derived from MTAL and 3-MOCA (with $\bar{\mathscr{A}} = \mathscr{A}$). The resulting model will be referred to as 3-MT-MO. The constraints where MTAL and 3-MOCA interact are as follows:

$$\sum_{(a,b)\in\mathscr{A}^2:a,b\in Q} w_{ab} \geq \sum_{a\in Q} z_a - 3 \qquad Q \subseteq \mathscr{A}, |Q| = 4. \qquad (11.10)$$

Constraint (11.10) replaces (11.5b) in the integrated model.

11.8.2 AP Location and Maximum-Efficiency Channel Assignment

A difficulty in integrating AP location and maximum-efficiency channel assignment is that the sets $\mathscr{F}_i$ and $\mathscr{P}_i$ for TP i depend on the set of installed APs. In order to define an integrated model, we need to introduce additional variables to represent this relation, and additional notation to specify the input data. For simplicity, we will consider 3-MECA with three non-overlapping channels. A similar integration can be obtained for the more general case.

Let $\tilde{\mathscr{A}}_i$ be the set of APs within the carrier sense range of TP i. We denote by $\mathscr{Q}_i$ the set of TPs within the carrier sense range of TP i, and by $\mathscr{R}_i$ the set of TPs that are covered by some AP $a \in \tilde{A}_i$, excluding i and those in $\mathscr{Q}_i$. Note that $\mathscr{Q}_i$ is the set of TPs that are potential direct interferers to i, while $\mathscr{R}_i$ is the set of TPs to which i is potentially exposed. Which users in $\mathscr{Q}_i$ and $\mathscr{R}_i$ will contend for the medium with i depends on the association (i.e., on the AP location) and the channel assignment.

We introduce the following variables:

$$\omega_{ih} = \begin{cases} 1 & \text{if TPs } i,h \in \mathscr{I} \text{ operate on the same channel,} \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{ih} = \begin{cases} 1 & \text{if TPs } i,h \in \mathscr{I} \text{ contend for the medium,} \\ 0 & \text{otherwise.} \end{cases}$$

In contrast to 3-MECA, explicit variables ω_{ih} are necessary because the associations $\bar{a}(i)$ and $\bar{a}(h)$ are not part of the input. We obtain the integrated model 3-MT-ME:

$$\max \quad \frac{1}{I} \sum_{i \in \mathscr{I}} \sum_{a \in \mathscr{A}_i} \frac{t_{ai} x_{ai}}{1 + \sum_{h \in \mathscr{I}} y_{ih}} \tag{11.11a}$$

$$\text{s.t.} \quad (11.2b), (11.2c), (11.2d), (11.2e), (11.2f)$$

$$(11.5c), (11.5d), (11.5e), (11.5f)$$

$$(11.10)$$

$$z_a + \sum_{b \in \mathscr{A}_i : t_{ai} > t_{bi}} x_{bi} \leq 1 \qquad\qquad i \in \mathscr{I}, a \in \mathscr{A}_i \qquad (11.11b)$$

$$\omega_{ih} = \sum_{a \in \mathscr{A}_i} \sum_{b \in \mathscr{A}_h} x_{ai} x_{bh} w_{ab} \qquad\qquad i, h \in \mathscr{I} \qquad (11.11c)$$

$$y_{ih} \geq \omega_{ih} \qquad\qquad i \in \mathscr{I}, h \in \mathscr{Q}_i \qquad (11.11d)$$

$$y_{ih} \geq \sum_{a \in \mathscr{A}_i \cap \mathscr{A}_h} \omega_{ih} x_{ah} \qquad\qquad i \in \mathscr{I}, h \in \mathscr{R}_i \qquad (11.11e)$$

$$y_{ih} \in \{0, 1\} \qquad\qquad i, h \in \mathscr{I}, i \neq h$$

$$\omega_{ih} \in \{0, 1\} \qquad\qquad i, h \in \mathscr{I}, i \neq h.$$

In 3-MT-ME, the objective function (11.11a) is the efficiency of AP location and channel assignment. Constraints (11.11b) state that if an AP a covering a TP i is installed, then i is not associated with an AP b providing a lower rate than a. Note that these constraints are not required in model MTAL because of its objective function. Constraints (11.11c) define variables ω_{ih}: Two TPs i and h operate on the same channel if and only if they are associated with two APs a and b that are assigned the same channel. Constraints (11.11d) state that a TP h within carrier sense range of TP i contends for the medium (i.e., $y_{ih} = 1$) if TPs i and h operate on the same channel, thus defining condition 2 of medium contention (see Section 11.7). Constraints (11.11e) define condition 3, stating that a TP h to which i is exposed contends for the medium with i, if i and h operate on the same channel, and if h is associated with an AP within carrier sense range of i. Since the carrier sense range is at least as large as the coverage range, constraints (11.11e) define condition 1 as well.

Problem 3-MT-ME is very complex, due to the hyperbolic objective function and to the large number of variables and constraints, some nonlinear. Exact and heuristic solution approaches to this problem are currently under study by some of the authors, but their description is beyond the scope of this chapter.

11.9 Experimental Results

We present computational experiments for a network instance representing a part of the WLAN deployed at Zuse Institute Berlin. There are 32 candidate AP locations and 798 TPs distributed on two floors, as shown in Figure 11.5. All APs are of type Cisco AP-1200/AP21G [7] and compliant with the IEEE 802.11g standard. Path loss predictions are obtained by 3D ray-tracing methods with multiple reflections using a 3D model of the building [19, 20].
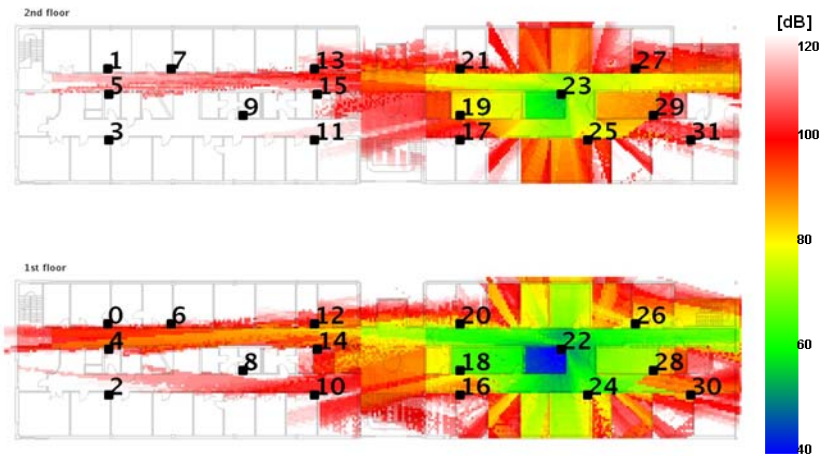
Fig. 11.5 Candidate APs locations and pathloss predictions for AP location 22

For comparison, we define a reference planning solution, which has been deployed in this network. In the reference solution there are eight installed APs using the three non-overlapping channels 1, 6, and 11, and the additional channel 7. The APs are installed in service rooms with thick concrete walls, located at the center of the corridors, resulting in a rather low throughput at many TPs. In particular, single-user throughput is below 1 Mbit/s at 22.75% of the TPs and the coverage loss is 11.5%. Figures 11.6(a), 11.7(a), and 11.9(a) show respectively single-user throughput, channel overlap, and efficiency for the reference solution. The thickness of lines in Figure 11.7(a) represents the amount of overlap. Note that not all TPs are covered in the reference solution. Indeed, by applying a minimum-cardinality set covering model, it is revealed that at least nine APs are required for full coverage. To make the comparisons meaningful, we set $M = 8$ as the budget for the AP location. Moreover, in order to apply and compare the minimum-overlap approach and the maximum-efficiency approach for channel assignment, in our experiments we use the three non-overlapping channels.
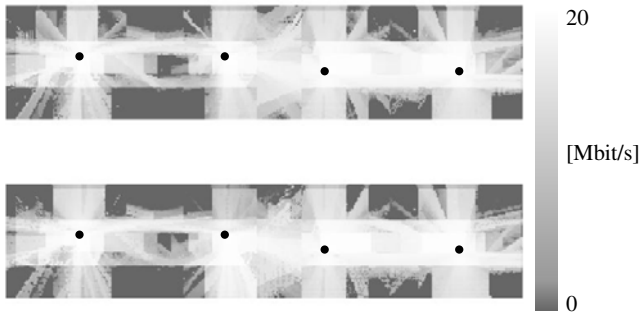
(a) Reference solution: the APs are located in the middle section of the
     building, leading to large low-throughput areas

(b) MTAL solution allows for maximum average throughput
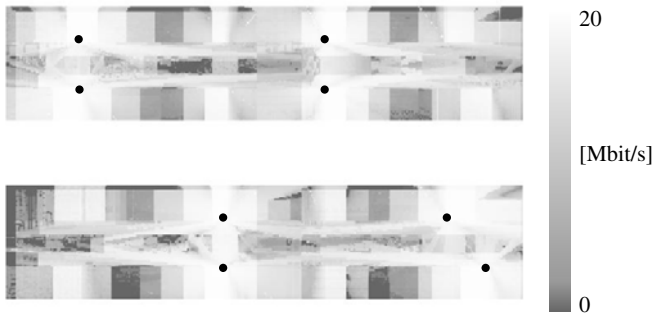
(c) 3-MT-MO solution ($\alpha = 0.6$): some throughput is sacrificed for re-
     ducing overlap

Fig. 11.6 Effect of optimization on single-user throughput
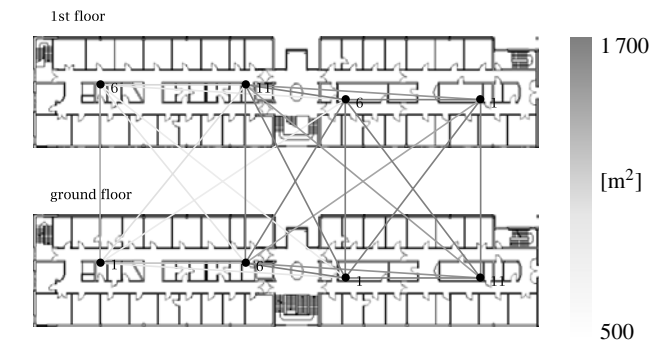
(a) Reference solution
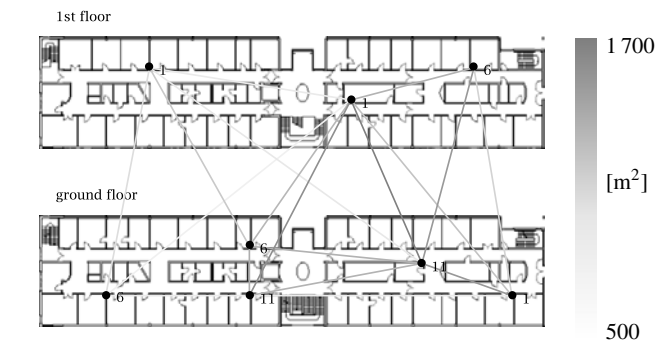
(b) Mᴛᴀʟ solution with 3-Mᴏᴄᴀ channel assignment: APs with large
pairwise overlap are assigned different channels
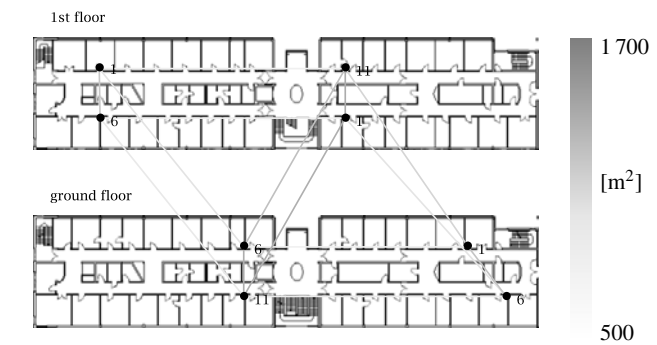
(c) 3-Mᴛ-Mᴏ solution ($\alpha = 0.6$): the AP positioning has very little
overlap to begin with, and the channel assignment further reduces it

Fig. 11.7 Overlap graph and channel assignment

(a) Channel assignment for minimum overlap

(b) Channel assignment for maximum efficiency: a slight change enables higher efficiency in the service areas of some APs
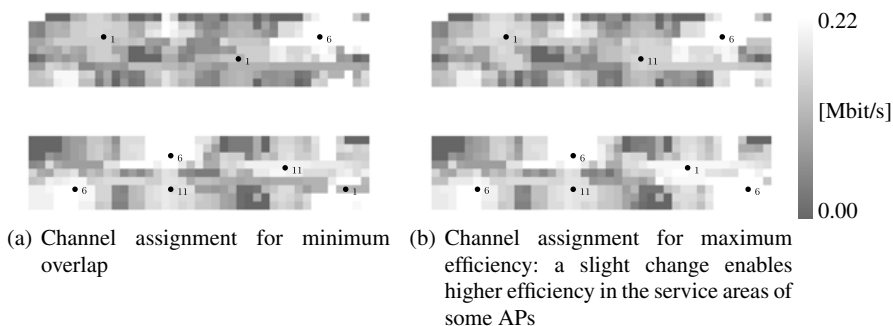
Fig. 11.8 Channel assignment by optimizing overlap vs. efficiency, evaluated in the latter metric (APs are positioned for maximum throughput)

(a) Reference configuration: many test points have low efficiency

(b) Integrated optimization makes for better efficiency
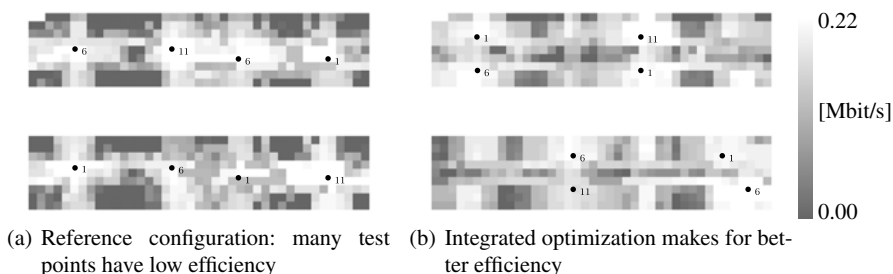
Fig. 11.9 Effect of integrated optimization on efficiency

The experiments are organized as follows. First, model MTAL is applied to find a solution of AP location that maximizes the single-user throughput. Then channel assignment is performed, with both the minimum-overlap approach (3-MOCA) and the maximum-efficiency approach (3-MECA). Finally, we illustrate integrated AP location and channel assignment by model 3-MT-MO.

Model MTAL is implemented using ZIMPL [22] and solved with ILOG CPLEX 10.0 [35]. The computational time is a few seconds. The single-user throughput of the optimal solution to MTAL is shown in Figure 11.6(b). It is possible to observe that each AP in this solution serves some areas on both floors, while in the reference solution the service area of each AP is mostly limited to the floor where the AP is installed. The total coverage loss in the optimized solution is only 1.71%. There is a significant improvement in the single-user throughput distribution in comparison to the reference solution. The average throughput is 23.11% higher, and the size of areas that either are not covered or suffer from low throughput (below 1 Mbit/s) is reduced by a factor of 3.5.

Table 11.1 Performance evaluation of the optimization results. For each network plan we evaluate the single-user throughput, the co-channel overlap, the pairwise co-channel overlap (MOCA objective function), and the network efficiency (MECA objective function)

Model	Throughput [Mbit/s]	Overlap [%]	Pairwise overlap (MOCA)	Efficiency (MECA)
Reference solution	10.69	48.93	14.38	89.55
MTAL + 3-MOCA	13.16	37.55	7.03	96.93
MTAL + 3-MECA	13.16	44.08	7.39	102.82
3-MT-MO, $\alpha = 0.1, 0.2$	13.16	31.56	2.11	103.13
3-MT-MO, $\alpha = 0.3, 0.4, 0.5$	13.10	21.41	1.39	107.17
3-MT-MO, $\alpha = 0.6, 0.7$	13.06	20.59	1.23	106.09
3-MT-MO, $\alpha = 0.8$	12.66	18.30	0.95	110.98
3-MT-MO, $\alpha = 0.9$	12.41	18.37	0.87	108.68
3-MT-MO, $\alpha = 1.0$	12.19	24.06	0.87	104.08

The next step is channel assignment. In Figure 11.7(b) we show the channel assignment obtained by solving model 3-MOCA, reporting its overlap graph (which is optimal for the given AP location), and in Figure 11.8(a) its evaluation in terms of the efficiency metric. For this network instance model, 3-MOCA can be solved very rapidly (in less than one second). Comparing the overlap graph of this solution with that of the reference solution (Figure 11.7(a)), it is easy to see that the former contains fewer links with high overlap. The improvement upon the reference solution in the average efficiency is 6.29%, although much of this improvement is due to the higher single-user throughput of the AP location obtained with model MTAL.

Figure 11.8(b) reports the efficiency metric evaluation for the channel assignment obtained by solving model 3-MECA. The model can be solved for this instance in a few seconds. The improvement in efficiency with respect to the channel assignment obtained with 3-MOCA is 6.07%, with an increase in overlap of 5.08%. The improvement in efficiency with respect to the reference solution is 12.75%.

We end the presentation of computational experiments by considering integrated AP location and channel assignment using model 3-MT-MO. We let α vary in the interval $(0, 1)$ with step size 0.1. Small values of α emphasize throughput maximization, while values close to 1.0 put more weight on overlap minimization. Note also that for a sufficiently small value of α the problem becomes equivalent to solving MTAL and 3-MOCA in two sequential steps.

In Table 11.1 we compare the reference solution to the results obtained from the models MTAL, 3-MOCA, 3-MECA, as well as from the combined model 3-MT-MO. The throughput column shows the single-user throughput averaged over the TPs that are within the serving range of at least one AP. The overlap performance metric is the fraction (in %) of the total area where the serving AP overlaps with at least one other AP from which the received signal is above the carrier sense threshold. Note that the latter is a performance evaluation metric and has not been taken into account explicitly in the optimization models. The last two columns show values of the two objective functions (MOCA and MECA) for each of the solutions.

From Table 11.1, we observe that, as for two-step optimization, the solution of the integrated model 3-Mт-Mо outperforms the reference solution for all values of α. The improvement in throughput declines when α grows, but the degradation is significant only if α is very close to 1. Up to $\alpha = 0.7$, throughput is hardly sacrificed when pursuing the goal of reducing overlap. The improvement over the two-step optimization is significant. The loss in throughput is very small, while the gain in both overlap and efficiency (even if 3-Mт-Mо is not designed to optimize efficiency) is remarkable, as it can be appreciated from Figure 11.9. This observation supports applying the integrated optimization to WLAN planning, whenever the computational complexity required can be afforded, and using the two-step optimization when a lower computational effort is required.

From Table 11.1 we conclude that $\alpha = 0.6$ appears as a good trade-off between the two objectives. The single-user throughput for the corresponding solution is shown in Figure 11.6(c), while the overlap map and the efficiency evaluation are shown respectively in Figures 11.7(c) and 11.9(b).

11.10 Conclusions and Perspectives

Designing wireless LANs for seamless coverage and high capacity in large office buildings is a challenge. On the one hand, good coverage and high throughput ideally call for a high density of APs. On the other hand, the availability of only three non-overlapping channels (out of 13 in total) suggests placing APs far apart. Other constraints and considerations, such as that APs typically need a wired network connection and that they should not be too easily accessible to prevent manipulation or damage, further complicate the planning task.

The challenge has been addressed in this chapter by means of mathematical optimization. The two aspects, providing coverage and controlling interference, can be addressed either in a two-step approach solving AP location and channel assignment separately, or with an integrated approach. These approaches have been compared in a real-life planning scenario for two adjacent floors of an office building. The manual design of the network, used as a reference, employs the same AP location pattern on both floors. Interference is limited by using four, partially overlapping channels. The optimized solutions clearly reveal a strong dependency among the AP locations and channel assignments across the two floors. Moreover, optimization significantly improves the coverage.

Comparing the optimization results with respect to the overlap of AP service areas using the same channel, solutions from the combined model are superior to those from the two-step approach. The difference, however, is not large enough to clearly justify the use of the much larger and more complex combined model. An appropriate trade-off between network performance and problem tractability must therefore be decided according to the needs and will change from case to case.

The network efficiency measure, which aggregates throughput and overlap, turns out to be quite consistent with the pairwise overlap measure in terms of the result-

ing networks. Indeed, the channel assignment of the solutions obtained from the combined model for alpha greater than or equal to 0.3 are optimal for the network efficiency measure. A more precise comparison of the networks resulting from these alternative objective functions would, however, require us to consider more network performance parameters.

AP service area overlap is used as a measure of interference in some of the channel assignment models. The actual interference situation cannot be properly compared on the basis of the mathematical models presented here. In fact, the effects of using overlapping channels in the reference design need to be assessed in a detailed simulation study, which is beyond the scope of this chapter.

In case overlapping channels can safely be deployed under sufficiently general conditions, the presented models are applicable, although at a higher computational price. When increasing the number of channels from three to as many as 13, specialized solution methods as well as heuristics are likely to be more successful than the plain application of an MIP solver.

References

1. Aardal, K. I., Hoesel, C. P. M. v., Koster, A. M. C. A., Mannino, C., Sassano, A.: Models and solution techniques for frequency assignment problems. Annals of Operations Research **153**(1), 79–129 (2007)
2. Aardal, K. I., Hurkens, C., Lenstra, J. K., Tiourine, S.: Algorithms for the radio link frequency assignment problem: The CALMA project. Operations Research **50**, 968–980 (2002)
3. Abusubaih, M., Gross, J., Wiethoelter, S., Wolisz, A.: On access point selection in IEEE 802.11 wireless local area networks. In: Proc. of the Sixth International Workshop on Wireless Local Networks (WLN 2006) (2006)
4. Amaldi, E., Bosio, S., Malucelli, F., Yuan, D.: On a new class of set covering problems arising in WLAN design. In: Proc. of the Intl. Network Optimization Conference (INOC '05), pp. 470–478 (2005)
5. Amaldi, E., Capone, A., Cesana, M., Malucelli, F.: Optimizing WLAN radio coverage. In: Proc. of the 2004 IEEE International Conference on Communications, vol. 1, pp. 180–184 (2004)
6. Bosio, S., Capone, A., Cesana, M.: Radio planning of Wireless Local Area Networks. IEEE/ACM Transactions on Networking **15**(6), 1414–1427 (2007)
7. Cisco Systems, Inc., http://www.cisco.com/en/US/products/hw/wireless/ps430/: Cisco Aironet 1200 Series Access Points, Data sheet
8. Cisco Systems, Inc.: Channel deployment issues for 2.4-GHz 802.11 WLANs. Tech. rep., http://www.cisco.com/ (2004)
9. Ehrgott, M.: Multicriteria Optimization, 2nd edn. Springer (2005)
10. Eisenblätter, A.: Frequency assignment in GSM networks: Models, heuristics, and lower bounds. Ph.D. thesis, Technische Universität Berlin, Berlin, Germany (2001)
11. Eisenblätter, A., Geerdes, H. F., Siomina, I.: Integrated access point placement and channel assignment for Wireless LANs in an indoor office environment. In: Proc. of the 8th IEEE Intl.

Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007) (2007)

12. Eisenblätter, A., Grötschel, M., Koster, A. M. C. A.: Frequency planning and ramifications of coloring. Discussiones Mathematicae Graph Theory **22**(1), 51–88 (2002)

13. Hansen, P., de Aragão, M. V. P., Ribeiro, C. C.: Boolean query optimization and the 0-1 hyperbolic sum problem. Annals of Mathematics and Artificial Intelligence **1**, 97–109 (1990)

14. Hills, A.: Large-scale wireless LAN design. IEEE Communications Magazine **39**(11), 98–107 (2001)

15. Hills, A., Friday, B.: Radio resource management in wireless LANs. IEEE Radio Communications Magazine **42**(12), S9–14 (2004)

16. IEEE Standards Association: IEEE Std 802.11-2007. `http://standards.ieee.org/getieee802/802.11.html` (2007)

17. Jaffrès-Runser, K., Gorce, J. M., Ubéda, S.: QoS constrained wireless LAN optimization within a multiobjective framework. IEEE Wireless Communications **13**(6), 26–33 (2006)

18. Jeavons, P., Dunkin, N., Bater, J.: Why higher order constraints are necessary to model frequency assignment problems. In: ECAI'98 Workshop on Non-binary constraints (1998)

19. Jemai, J., Piesiewicz, R., Kürner, T.: Calibration of an indoor radio propagation prediction model at 2.4 GHz by measurements of the IEEE 802.11b preamble (2002). COST 273 TD, Duisburg, Germany

20. Jemai, J., Reimers, U.: Channel modeling for in-home wireless networks. In: Proc. of IEEE Intl. Symposium on Consumer Electronics (ISCE02), pp. F123–F129. Erfurt, Germany (2002)

21. Kamenetsky, M., Unbehaun, M.: Coverage planning for outdoor wireless LAN. In: Proc. of Intl. Zurich Seminar on Broadband Communications, 2002. Access, Transmission, Networking (IZS). Zurich, Switzerland (2002)

22. Koch, T.: Rapid mathematical programming. Ph.D. thesis, TU Berlin, Germany (2004). Available at `http://www.zib.de/Publications/abstracts/ZR-04-58/`, ZIMPL is available at `http://www.zib.de/koch/zimpl`

23. Koster, A. M. C. A.: Frequency assignment – models and algorithms. Ph.D. thesis, Maastricht University (1999)

24. Kumar, A., Kumar, V.: Optimal Association of Stations and APs in an IEEE 802.11 WLAN. In: Proc. National Conference on Communications (NCC). India (2005)

25. Lee, Y., Kim, K., Choi, Y.: Optimization of AP placement and channel assignment in wireless LANs. In: Proc. of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02) (2002)

26. Leung, K. K., Kim, B. J.: Frequency assignment for IEEE 802.11 wireless networks. In: Proc. of the 58th IEEE Vehicular Technology Conference (VTC2003-Fall). Orlando, FL (2003)

27. Ling, X., Yeung, K. L.: Joint access point placement and channel assignment for 802.11 wireless LANs. In: Proc. of IEEE Wireless Communications and Networking Conference (WCNC 2005). New Orleans, LA (2005)

28. Mateus, G. R., Loureiro, A. A. F., Rodrigues, R. C.: Optimal network design for wireless local area network. Annals of Operations Research **106**(331-345) (2001)

29. Mishra, A., Banerjee, S., Arbaugh, W.: Weighted coloring based channel assignment for WLANs. ACM SIGMOBILE Mobile Computing and Communications Review **9**(3), 19–31 (2005)

30. Prommak, C., Kabara, J., Tipper, D., Charnsripinyo, C.: Next generation wireless LAN system design. In: Proc. of the IEEE Military Conference (MILCOM 2002), vol. 1, pp. 473–477 (2002)

31. Riihijärvi, J., Petrova, M., Mähönen, P.: Frequency allocation for WLANs using graph colouring techniques. In: Proc. of the Second Annual Conference on Wireless On-Demand Network Systemsand Services (WONS '05). St. Moritz, Switzerland (2005)

32. Sherali, H. D., Pendyala, C. M., Rappaport, T. S.: Optimal location of transmitters for microcellular radio communication system design. IEEE Journal on Selected Areas in Communications **14**(4), 662–673 (1996)

33. Siomina, I.: Wireless LANs planning and optimization. STSM Technical Report, COST Action TIST 293 (2005)

34. Siomina, I., Yuan, D.: Optimization of channel assignment and access point transmit power for minimizing contention in Wireless LANs. In: Proc. of the 5th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2007) (2007)
35. ILOG, Inc: ILOG CPLEX 10.0, User's manual (2006)
36. Unbehaun, M., Kamenetky, M.: On the deployment of picocellular wireless infrastructure. IEEE Wireless Communications **10** (2003)
37. Villegas, E. G., Ferré, R. V., Aspas, J. P.: Implementation of a distributed dynamic channel assignment mechanism for IEEE 802.11 networks. In: Proc. of the 16th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2005), vol. 3, pp. 1458–1462 (2005)
38. Villegas, E. G., López-Aguilera, E., Ferré, R. V., Aspas, J. P.: Effect of adjacent-channel interference in IEEE 802.11 WLANs. In: Proc. of the 2nd Intl. Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom '07) (2007)
39. Wertz, P., Sauter, M., Wölfle, G., Hoppe, R., Landstorfer, F. M.: Automatic optimization algorithms for the planning of wireless local area networks. In: Proc. of the 60th IEEE Vehicular Technology Conference (VTC2004-Fall). Los Angeles, CA (2004)
40. Zdarsky, F. A., Martinovic, I., Schmitt, J. B.: On lower bounds for MAC layer contention in CSMA/CA-based wireless networks. In: Proc. of Workshop on Discrete Algothrithms and Methods for MOBILE Computing and Communications (DIALM'05), pp. 8–16. Cologne, Germany (2005)

Chapter 12
Time-Efficient Broadcast in Radio Networks

David Peleg and Tomasz Radzik

Abstract Broadcasting is a basic network communication task, where a message initially held by a source node has to be disseminated to all other nodes in the network. Fast algorithms for broadcasting in radio networks have been studied in a wide variety of different models and under different requirements. Some of the main parameters giving rise to the different variants of the problem are the accessibility of knowledge about the network topology, the availability of collision detection mechanisms, the wake-up mode, the topology classes considered, and the use of randomness. This chapter introduces the problem, reviews the literature on time-efficient broadcasting algorithms for radio networks under a variety of models and assumptions, and illustrates some of the basic techniques.

Key words: radio networks, broadcasting, broadcasting sequence, selective families, unit disk graphs

12.1 Introduction

12.1.1 The Problem

A *radio network* consists of nodes, each equipped with a device enabling it to transmit and receive messages. At any given time, each node decides whether to act as a *transmitter* or as a *receiver*. Reception conditions are modeled by a network con-

David Peleg
Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel, e-mail: `david.peleg@weizmann.ac.il`

Tomasz Radzik
Department of Computer Science, King's College London, London WC2R 2LS, United Kingdom, e-mail: `tomasz.radzik@kcl.ac.uk`

necting the nodes, where the existence of a (directed) edge from a node v to a node u indicates that transmissions of v can reach u directly.

A node acting as a transmitter in a given time step transmits a message that reaches all of its outgoing neighbors in the network in the same time step. However, its being within range of a transmitting node does not necessarily ensure that its transmitted message is received successfully. A node acting as a receiver in a given step successfully *hears* a message if and only if exactly one of its incoming neighbors transmits in this step. If two or more incoming neighbors v and v' of u transmit simultaneously in a given step, then neither of their messages is heard by u. In this case we say that a *collision* occurred at u.

This chapter considers *broadcasting* in radio networks, which is the following basic communication task. Initially, one distinguished node s, called the *source*, has a message M that has to be delivered to all other nodes in the network. Remote nodes, which cannot receive the transmissions of s directly, have to get M via intermediate nodes.

The model considered in most of the literature on broadcasting algorithms in radio networks is synchronous. All nodes have individual clocks that tick at the same rate, measuring time steps, also referred to as *rounds*. Any execution of a broadcasting operation can be described as a sequence $\langle T_1, \ldots, T_t \rangle$, hereafter referred to as a *broadcasting sequence*, where each T_i is a *transmission set* consisting of the nodes that acted as transmitters in time step i. The execution time of a broadcasting algorithm in a given radio network is the number of rounds it takes since the first transmission until all nodes of the network hear the source message, or in other words, the length of the corresponding broadcast sequence.

The current chapter reviews the literature on time-efficient broadcasting algorithms for radio networks under a variety of models and assumptions.

The question of time-efficient broadcasting has been examined from two related but distinct viewpoints. The first, more mathematical, viewpoint concerns investigating the absolute optimum time required for broadcasting. For a radio network G, let $b(G)$ denote the length of the shortest possible execution of the broadcasting operation for G. The study of broadcasting time from this viewpoint concentrated on the abstract question of identifying or tightly bounding the function b, and thus focused on establishing the existence or nonexistence of short broadcasting sequences for a given network, using a variety of algorithmic, probabilistic, and nonconstructive methods.

To bound the function $b(G)$, it was necessary to identify the graph-theoretic parameters governing its behavior. Clearly, the *radius* of a network G from the source s, denoted $R(G,s)$ (i.e., the largest distance between s and any other node in G), serves as a lower bound for the length of any broadcasting sequence on G. For uniformity over the different sources, we use the network diameter D (i.e., the largest distance between any two nodes in G) instead. (Note that $D/2 \leq R(G,s) \leq D$ for every node s in G.) It also stands to reason that n, the number of nodes in the network, plays a role in the cost of broadcasting. Consequently, let us define $\hat{b}(n,D) = \max b(G)$, where the maximum is taken over all radio networks G of n

nodes and diameter D. The mathematical problem stated above now becomes identifying the function $\hat{b}(n, D)$.

The second viewpoint deals with the more algorithmic question of developing protocols for time-efficient broadcasting in realistic settings. This question can be studied in a variety of models. To begin, one may consider a centralized setting. In this setting, we assume that the graph topology is known in advance, and the goal is to design, in an off-line preprocessing stage, an optimal (time-minimal) or near-optimal solution for the broadcasting problem. Such a solution can sometimes be represented in the form of a fixed *schedule*, which essentially describes the optimal execution, i.e., specifies a broadcasting sequence $\langle T_1, \ldots, T_t \rangle$. The schedule can be applied as a broadcast procedure in the natural way: In step i, every node $v \in T_i$ which already holds a copy of the source message M transmits it. A node $v \in T_i$ that does not have a copy yet remains silent. The schedule S is a *broadcast schedule* for the source s in G if after applying it, every node in the network has a copy of M.

While the question of determining the minimum length of broadcasting schedules for radio networks is very basic and of considerable theoretical interest, in reality such schedules are hardly ever used, for a variety of practical reasons. For instance, such centrally computed fixed schedules are only useful when the network is relatively stable; in dynamic environments, where the network topology constantly changes, such schedules are hard to construct, maintain, and update.

The common practice is therefore to employ distributed broadcasting protocols, which generate the broadcasting sequence "on the fly," while performing the broadcasting operation itself. This approach has obvious advantages in terms of simplicity and flexibility. On the down side, the typical situation in the distributed context is that nodes are unaware of the topology of the network and have limited or no knowledge of other parameters of the network, such as its diameter or size. They may not even be aware of their immediate neighborhood, namely, the identity of their neighbors. Networks in which the nodes have such limited knowledge are often called *ad hoc* networks. This lack of centralized knowledge rules out the efficient design of optimal or even near-optimal execution sequences. In fact, distributed broadcasting algorithms that rely on partial knowledge of the topology must usually involve trial-and-error, possibly incurring wasteful (but unavoidable) collisions, and are generally expected to generate significantly slower broadcasting sequences. Hence the focus in this area is on developing efficient algorithms that quickly adapt to the network at hand, possibly through learning its topology or some of its properties, and manage to tailor a relatively short execution sequence for it. Let $\hat{b}_{DD}(n, D)$ (respectively, $\hat{b}_{DR}(n, D)$) denote the time required for broadcasting on an n-node radio network of diameter D by a distributed deterministic (or randomized) algorithm. (Clearly, this function depends on the precise model and class of networks under consideration.)

Interestingly, despite the obvious advantages of adaptivity, which advocate the use of fully distributed, local, online, and dynamically adapting solutions, in some cases the best algorithms currently known are essentially *oblivious*, in the sense that the actions taken by each node depend only on its identity and the current time step, but not on the history of the execution. Such algorithms can be thought of as requiring the nodes to follow fixed precomputed schedules, designed by a central

authority. Note, though, that there is an inherent difference between these oblivious schedules and the ones mentioned earlier. Our previous discussion concerned individualized schedules, especially tailored for every given radio network separately, relying on a complete knowledge of its topology. In contrast, the approach of using oblivious schedules provides a *general* distributed solution, applicable in situations where no knowledge of the network topology is available. In particular, this implies that the oblivious schedules provided by the construction algorithm must be *universal*, i.e., efficiently applicable for broadcasting on *every* network, regardless of its topology.

A well-studied technique developed for constructing such universal schedules is based on the combinatorial notion of *selection sequences* and *selective families* of transmission sequences for broadcasting. Subsequently, selective families, and the related structures of strongly selective families, cover-free families, superimposed codes, and selectors, were used for broadcasting as well as for other communication tasks in radio networks, such as wakeup and synchronization.

12.1.2 Model Parameters

The complexity of broadcasting in radio networks critically depends on the particular setting and model parameters, and may change significantly depending on whether or not the nodes know the network, how their actions are coordinated, what mechanisms are available to the network nodes, and so on. Let us give a brief overview of the main parameters affecting the performance of broadcasting algorithms in radio networks.

Collision detection. In the event of a collision of two or more transmissions, one of a number of possibilities might occur at each of the receiving nodes. Three main alternatives are the following.

> (C1) The receiving node hears nothing, and cannot tell if a collision occurred or none of its neighbors transmitted, i.e., it cannot distinguish a collision from silence.
> (C2) The receiving node detects the fact that a collision has occurred.
> (C3) The signal of exactly one of the transmitted messages prevails, and that message is received correctly by the receiving node.

The most commonly studied model in the literature on broadcasting algorithms in radio networks is the "collision-as-silence" model, which assumes that possibility (C1) always occurs, namely, the effect of a collision is the same as that of a step in which no transmissions took place. Certain papers consider the alternative *collision detection* model, which assumes that possibility (C2) always occurs, namely, the receiving nodes always recognize collisions. A third model, which is just as reasonable but has received even less attention, is the *flaky collision* model, in which at any receiver, the result of a collision may be either of the possibilities (C1) and (C3), i.e., some collisions result in one message getting through, while some others

result in the effect of silence. One may envision an even weaker variant of the flaky model, again rather reasonable, where the outcome of a collision could be any of three possibilities. In practice, more elaborate models can be considered, such as different interference and transmission ranges, or models allowing for the availability of carrier sensing mechanisms.

Wakeup model. The way the nodes join the broadcasting process can be modeled in two different ways. Most of the existing literature on broadcasting algorithms in radio networks considers the *conditional wakeup* model, where the nodes other than the source are initially idle and cannot transmit until they receive the source message for the first time and subsequently wake up. Namely, the clock of a node starts in the round when the node first receives the source message (the clock of the source starts at the beginning of the execution). One may assume without loss of generality that the number of rounds that have passed since the beginning of the execution is appended to every message, so that all nodes can emulate the source's clock.

Alternatively, one may consider the *spontaneous wakeup* model, where all nodes are assumed to be awake when the source transmits for the first time, and may contribute to the broadcasting process by transmitting control messages even before they receive the source message. In other words, the clocks of all nodes start simultaneously, in the round when the source transmits for the first time. In this model, nodes far away from the source can utilize their waiting time to perform some preprocessing stage during which they can gather necessary information and possibly construct some auxiliary structures in the network (a sparse spanner, for example), facilitating faster message propagation at a later stage.

The task of broadcasting in the conditional wakeup model can in fact be interpreted as activating the network from a single source, and is related to the task of waking up the network. In this latter task, some nodes spontaneously wake up and have to wake up other nodes by sending messages. Thus, broadcasting in the conditional wakeup model, i.e., activating the network from a single source, is equivalent to waking up the network when exactly one node (the source) wakes up spontaneously. The broadcasting models with spontaneous wakeup and conditional wakeup have also been called broadcasting with and without spontaneous transmissions, respectively.

Directionality. Most papers in the area assume that the radio network at hand is undirected, i.e., for every two nodes u and v, if v is within range of u's transmissions then u is within range of v as well. However, some papers consider also a model for radio networks based on a *directed* graph, allowing us to model asymmetric situations. (Such asymmetry can result from topographic conditions or from differences in the strength of the transmission equipment at different nodes.) Throughout most of what follows we will consider undirected graphs, except where mentioned otherwise.

Graph classes. In addition to the study of general (arbitrary topology) radio networks, some recent interest arose concerning a special subclass, referred to as *UDG radio networks*, where the network is modeled as a *unit disk graph* (UDG) whose nodes are represented as points in the Euclidean plane. Two nodes are joined by an

edge if their distance is at most 1. The node transmitters have power enabling them to transmit to distance 1.

Randomness. Both deterministic and randomized broadcasting algorithms were considered in the literature. Randomized algorithms can also be oblivious, in which case the actions of the nodes depend on their identity and the time as well as on the outcomes of their random choices, but not on the execution history.

The harsh model. In most of the literature on algorithms for radio networks, the effects of transmission collisions are modeled by the collision-as-silence model, and the wakeup pattern of the network nodes is assumed to follow the conditional wakeup model. Consequently, we will focus on a model assuming both conditional wakeup and collision-as-silence, except where mentioned otherwise. We hereafter refer to this model as the *harsh* model for radio networks. Note that it is typically easier to establish lower bounds on broadcasting time in this model, although the resulting bounds will not apply to the other models. On the other hand, algorithms working in the harsh model will clearly apply also in models supporting collision detection or allowing spontaneous wakeup.

12.2 Efficient Schedules and Bounds on $b(G)$ and $\hat{b}(n,D)$

The literature concerning algorithmic aspects of radio broadcasting can be divided into two subareas, one dealing with centralized communication, in which it is assumed that nodes have complete knowledge of the network topology, and hence can simulate a central transmission scheduler (cf. [1, 23–25, 49, 58, 62, 81]), and the other assuming only limited, usually local, knowledge of topology and studying distributed communication in such networks.

The first papers to formulate the abstract model of radio networks and address the question of broadcasting in radio networks were [23, 24]. These papers described methods for the centralized design of a broadcasting schedule in radio networks, assuming complete knowledge of the network topology. In [23], it was also shown that the problem of finding a shortest broadcasting schedule (or alternatively, computing the minimum broadcasting time $b(G)$) for an arbitrary input graph G is NP-hard. As often happens, this hardness result has led to the development of two research directions, concerning *global bounds* and *approximations*.

The study of global bounds on broadcasting time was initiated in [25], which established an upper bound of $\hat{b}(n,D) = \mathscr{O}(D\log^2(n/D))$ on broadcasting time. This upper bound was proved by presenting a deterministic algorithm for generating a broadcasting schedule of length $\mathscr{O}(D\log^2 n)$ for every n-node network of diameter D. Shortly afterwards, a randomized algorithm completing broadcasting in time $\mathscr{O}(D\log n + \log^2 n)$ with high probability was presented in [10]. This, in turn, implied that $\hat{b}(n,D) = \mathscr{O}(D\log n + \log^2 n)$. Note, however, that this proof of the upper bound on $\hat{b}(n,D)$ is probabilistic, or nonconstructive. A deterministic construction for such schedules of length $\mathscr{O}(D\log n + \log^2 n)$ was later presented in [77].

On the other hand, it was proved in [1] that there exists a family of n-node networks of radius 2, for which any broadcasting schedule requires $\Omega(\log^2 n)$ communication rounds. This lower bound holds even assuming that the nodes have complete knowledge of the network and there are no restrictions on the coordination mechanism. Given that the diameter D is also a lower bound on broadcasting time, as mentioned earlier, it followed that $\hat{b}(n,D) \geq D + \Omega(\log^2 n)$. Note, though, that there is a fine distinction between these two lower bounds. The diameter D is a *global* or *universal* lower bound, in the sense that $b(G) \geq D$ for every graph G of diameter D. In contrast, the $\Omega(\log^2 n)$ lower bound of [1] is *specific* or *existential*, in the sense that it only implies that $b(G) \geq \log^2 n$ for *some* low diameter graphs; clearly, there are also low diameter graphs G for which $b(G) \ll \log^2 n$, and in fact, there are $\mathcal{O}(1)$ diameter graphs G for which $b(G) = \mathcal{O}(1)$.

This gap between the upper and lower bounds established in [10] and [1] raised the intriguing question, formulated in [1, 83], of whether further pipelining may be possible, leading to a separation of the D and $\log n$ terms in the upper bound. This has motivated a succession of improvements of the upper bound on $\hat{b}(n,D)$. The separation was first achieved in [58], which established a bound of $\hat{b}(n,D) = D + \mathcal{O}(\log^5 n)$. This was again done via a nonconstructive proof, by presenting a randomized algorithm for constructing short broadcasting schedules (of length $D + \mathcal{O}(\log^5 n)$) with high probability. This algorithm is based on partitioning the underlying graph into low-diameter clusters and coloring them with $\mathcal{O}(\log n)$ colors, and subsequently constructing a broadcasting schedule in each cluster separately, by applying the construction algorithm of [10] as a subprocedure. (Using the algorithm of [25] as the subprocedure instead, the resulting construction algorithm is deterministic but the broadcasting schedule it constructs is of length $D + \mathcal{O}(\log^6 n)$.) The clustering method from [58] has next been improved in [49], which reduced the upper bound on $\hat{b}(n,D)$ to $D + \mathcal{O}(\log^4 n)$, again using a randomized algorithm and hence yielding a nonconstructive proof. (Using the algorithm of [25] as the subprocedure instead, the resulting algorithm is deterministic but the constructed broadcasting schedules are of length $D + \mathcal{O}(\log^5 n)$.)

Finally, the optimal bound of $D + \mathcal{O}(\log^2 n)$, yielding the sought $\hat{b}(n,D) = D + \Theta(\log^2 n)$, was established in [62]. This bound was once again established via a probabilistic (nonconstructive) proof, although based on a different construction method (whose deterministic version yielded only broadcasting schedules of length $D + \mathcal{O}(\log^3 n)$). That paper still left open the question of constructing broadcasting schedules of length $D + \mathcal{O}(\log^2 n)$ *deterministically*. While this question has not been completely answered yet, explicit constructions of broadcasting schedules of length $\mathcal{O}(D + \log^2 n)$ and $D + \mathcal{O}(\frac{\log^3 n}{\log\log n})$ were recently presented in [81] and [34] respectively.

The study of approximations for optimal broadcasting time has so far led mostly to negative results. It has been proved in [47] that approximating the broadcasting time $b(G)$ for arbitrary n-node network G by a multiplicative factor of $o(\log n)$ is impossible under the assumption that $NP \subseteq BPTIME(n^{\mathcal{O}(\log\log n)})$. Under the same assumption, it was also proved in [48] that there exists a constant c such that there is no polynomial-time algorithm which produces, for every n-node graph G, a broad-

casting schedule of length at most $b(G) + c\log^2 n$. (Naturally, since D is a global lower bound on $b(G)$, the algorithm of [25] can be thought of also as an approximation algorithm for $b(G)$, with ratio $\mathcal{O}(\log^2(n/D))$.)

12.3 Distributed Broadcasting Algorithms

12.3.1 Deterministic Distributed Algorithms

The study of distributed broadcasting in radio networks was initiated in [10]. The model studied therein assumed that nodes have limited knowledge of the topology, and specifically, that they know only their own identity and that of their neighbors. Under these assumptions, even in the harsh model, broadcasting can be achieved deterministically by a simple linear time algorithm based on depth-first search [4], establishing that $\hat{b}_{DD}(n,D) = \mathcal{O}(n)$. Obtaining a matching lower bound on $\hat{b}_{DD}(n,D)$ proved to be more elusive than previously anticipated. The highest lower bound to date is $\hat{b}_{DD}(n,D) = \Omega(n^{1/4})$, due to [78], where a class of graphs of diameter 4 is constructed, such that every broadcasting algorithm requires time $\Omega(n^{1/4})$ on at least one of these graphs. A linear lower bound, $\hat{b}_{DD}(n,D) = \Omega(n)$, was proved in [10] on a class of radio networks of diameter 3; however, it turns out that the proof applies only to the flaky collisions model. In the collision detection model, the question of establishing matching upper and lower bounds for distributed deterministic broadcasting was posed as an open problem in [10], and its status remains practically unchanged to date. (The best upper bound currently known in this model is still $\mathcal{O}(n)$; in addition, the problem has been resolved for some specific graph classes, as discussed later in Section 12.4.2.)

A number of subsequent papers [1, 18, 26, 28, 33, 37, 45, 80] studied deterministic distributed broadcasting in ad hoc radio networks, i.e., under the more strict assumption that nodes know only their own identities but not those of their neighbors, and that the topology of the network is unknown. A lower bound of $\hat{b}_{DD}(n,D) = \Omega(D\log n)$ on broadcasting time in the harsh model was proved in [18], and this lower bound was subsequently sharpened to $\hat{b}_{DD}(n,D) = \Omega(n\frac{\log n}{\log(n/D)})$ in [79]. The two fastest distributed deterministic algorithms to date in this model, presented in [42] and [79], achieve broadcasting in time $\mathcal{O}(n\log^2 D)$ and $\mathcal{O}(n\log n)$ respectively. The algorithm of [79] was constructed by first defining an algorithm in a model allowing collision detection, and then applying a technique for simulating collision detection in the collision-as-silence model, developed in [78]. In contrast, the algorithm of [42] is oblivious and makes use of efficient deterministic selection sequences. (Note, though, that this and similar deterministic algorithms are not "fully oblivious," in the sense that they do make use of global time information received from their neighbors upon starting.) Combining these two algorithms thus yields $\hat{b}_{DD}(n,D) = \mathcal{O}(n\min\{\log^2 D, \log n\})$, leaving a small gap between the best upper and lower bounds currently known.

The problem was considered also in the spontaneous wakeup model. A deterministic algorithm completing broadcast in time $\mathcal{O}(n)$ for arbitrary n-node networks is presented in [26]. For this model, a matching lower bound of $\hat{b}_{DD}(n, D) = \Omega(n)$ on deterministic broadcasting time, even for the class of networks of constant radius, was proved in [80], by an adaptation of the proof of [10].

Again, it is not clear whether allowing a collision detection mechanism can be used to improve broadcasting time. In particular, it is not known whether allowing both spontaneous wakeup and collision detection may lead to sublinear time broadcasting algorithms.

12.3.2 Randomized Distributed Algorithms

The first paper to study randomized distributed broadcasting algorithms in radio networks was [10]. The model does not assume knowledge of the network topology or availability of distinct identities. The paper presents a randomized broadcasting algorithm with expected time $\mathcal{O}(D\log n + \log^2 n)$.

Hence, in view of the lower bounds of [10, 78], there is an exponential gap between determinism and randomization in the time of radio broadcasting, in the collision-as-silence and flaky collisions models, for low diameter networks.

Lower bounds for randomized distributed broadcasting were given in [1, 83]. In [83] it was shown that for any (deterministic or randomized) broadcasting algorithm and parameters $D \leq n$, there exists an n-node network of diameter D requiring expected time $\Omega(D\log(n/D))$ to execute this algorithm, yielding a lower bound of $\hat{b}_{DR}(n, D) = \Omega(D\log(n/D))$ in the harsh model. The $\Omega(\log^2 n)$ lower bound of [1], for some networks of radius 2, rules out the existence of schedules shorter than $c\log^2 n$ for some constant $c > 0$; hence, it implies also that for randomized algorithms, $\hat{b}_{DR}(n, D) = \Omega(\log^2 n)$. Let us remark that this lower bound holds also in models with collision detection or spontaneous wakeup. Subsequently, randomized algorithms working in expected time $\mathcal{O}(D\log(n/D) + \log^2 n)$, and thus matching the above lower bounds, were obtained independently in [42, 79], establishing a tight bound of $\hat{b}_{DR}(n, D) = \Theta(D\log(n/D) + \log^2 n)$. These algorithms are oblivious, using techniques based on universal selection sequences, and subsequently they operate in the harsh model.

It is currently unclear what happens if the model is not harsh, and specifically, whether the upper bound can be improved in a model allowing collision detection, spontaneous wakeup, or both. The left part of the lower bound may possibly still hold even assuming collision detection or spontaneous wakeup. Conversely, the right part of the lower bound holds for schedule length; hence, it certainly holds in any distributed model, including non-harsh ones, and it is tight even in the harsh model, in view of the matching upper bound

12.3.3 Randomized Broadcasting: Main Techniques

We illustrate the main techniques used in randomized distributed broadcasting in ad hoc radio networks by describing and analyzing an $\mathcal{O}(D\log n + \log^2 n)$-time algorithm. This algorithm works for directed networks and is essentially the algorithm given in [10] but with some modifications suggested later (for example, in [42, 79]) to simplify analysis. Throughout this and the next subsections, a neighbor of a node v means an in-neighbor, that is, a node which can transmit to v.

The main issue in organizing communication in a radio network of unknown topology is breaking the symmetry which arises when a number of neighbors of a node v consider transmitting, not knowing about each other. Node v receives a message only if its neighbors manage to select exactly one of them to proceed with transmission. If nodes have access to random bits, then such a selection can be attempted in the following way. Each node which wants to transmit decides randomly, and independently of the decisions made by the other nodes, whether it should transmit. If the probabilities of transmitting are chosen appropriately, then there may be a good chance that exactly one neighbor of node v transmits (and node v receives a message).

As an example, assume that the nodes in the network which want to transmit include $k \geq 1$ neighbors $w_1, w_2, \ldots, w_k$ of node v. If during the current step each node which wants to transmit does so with probability $1/k$ (and remains silent with probability $1 - 1/k$), then node v receives a message with probability at least e^{-1}:

$$\text{Prob}(v \text{ receives a message})$$
$$= \text{Prob}(\text{exactly one node } w \in \{w_1, w_2, \ldots, w_k\} \text{ transmits})$$
$$= k\frac{1}{k}\left(1 - \frac{1}{k}\right)^{k-1} \geq e^{-1}.$$

The probability of transmission does not have to be exactly $1/k$ for node v to receive a message with positive constant probability. For example, node v receives a message with probability at least $1/4$, if each node which wants to transmit does so with probability $1/k'$ and $k'/2 \leq k \leq k'$:

$$\text{Prob}(v \text{ receives a message}) = k\frac{1}{k'}\left(1 - \frac{1}{k'}\right)^{k-1}$$
$$\geq \frac{k}{k'}\left[\left(1 - \frac{1}{k'}\right)^{k'}\right]^{k/k'} \geq \min_{1/2 \leq x \leq 1}\left\{x\left(\frac{1}{4}\right)^x\right\} \geq \frac{1}{4}. \qquad (12.1)$$

If the nodes do not have any estimate on how many of them may currently be competing to transmit to the same node (different nodes may have different numbers of neighbors which want to transmit), then the nodes can try different probabilities of transmission in different steps. Consider a *round* of $\lceil \log n \rceil$ steps given in Figure 12.1, where the probability of transmission decreases geometrically in each step. Node v receives a message during this round (from one of its neighbors $w_1, w_2, \ldots, w_k$) with probability at least $1/4$:

Prob(v receives a message in one round)

$\geq$ Prob(v receives a message in step $\lceil \log(k+1) \rceil$ of this round)

$$\geq \frac{1}{4}, \tag{12.2}$$

where the last inequality is Inequality (12.1) with $k' = 2^{\lceil \log(k+1) \rceil}$.

---

for each node v in parallel **do**
    if v wants to transmit in this round **then**
        for $i = 1$ to $\lceil \log n \rceil$ **do**
            v transmits with probability $1/2^i$

Fig. 12.1 One round of the randomized broadcasting algorithm

The round of $\lceil \log n \rceil$ steps given in Figure 12.1 is the basic tool in designing randomized distributed algorithms for broadcasting as well as for other communication tasks in radio networks. We consider algorithm RANDOMIZEDBROADCAST which is a sequence of such rounds. The *active nodes* are the nodes which have already received the source message. The nodes which want to transmit in the current round are the nodes which are active at the beginning of this round. An inactive node can only listen and becomes active when it receives a message. Note that when a node becomes active, it does not immediately join the computation but waits until the beginning of the next round (assume that messages contain the step count, so an activated node knows when the next round will start). This waiting is not essential, but simplifies the analysis. We now show a bound on the number of rounds needed to complete broadcasting.

Bounds on the running times of randomized algorithms are usually given either as bounds on the *expected* running time or as bounds which hold *with high probability* (w.h.p.), and "high probability" means normally a probability of at least $1 - 1/n$, where n is the size of input. For the RANDOMIZEDBROADCAST algorithm, these two types of bounds are asymptotically the same (as commonly happens with randomized algorithms). We first show that the number of rounds in our algorithm is $\mathcal{O}(D + \log n)$ w.h.p., and then we use this result to show that the expected number of rounds is of the same order. Each round consists of $\lceil \log n \rceil$ steps, so an $\mathcal{O}(D + \log n)$ bound on the number of rounds implies an $\mathcal{O}(D \log n + \log^2 n)$ bound on the total number of steps.

Let V denote the set of nodes and let T_v be the round when a node v becomes active. The broadcasting completes in $T = \max_{v \in V}\{T_v\}$ rounds. We show there is a constant c such that for each node $v \in V$,

$$\text{Prob}(T_v > c(d(v) + \log n)) \leq \frac{1}{n^2}, \tag{12.3}$$

where $d(v)$ is the shortest-path distance from the source node s to v. Inequality (12.3) implies that the number of rounds is $\mathcal{O}(D + \log n)$ w.h.p.:

$$\text{Prob}(T > c(D + \log n)) = \text{Prob}(T_v > c(D + \log n), \text{ for some } v \in V)$$

$$\leq \sum_{v \in V} \text{Prob}(T_v > c(d(v) + \log n)) \leq \frac{1}{n}.$$

To show that (12.3) holds, we take an arbitrary node $u \in V \setminus \{s\}$ other than the source and a shortest path $P = (s = v_0, v_1, v_2, \ldots, v_d = u)$ from s to u, where $d = d(u)$, and analyze the progression of the source message along this path. We view the sequence of rounds as a sequence $\mathscr{A}$ of *trials* to deliver a message to the nodes on this path *in the order* in which they appear on the path. Let a node v_i be the *waiting node* in the current round (node v_1 is the waiting node in round 1). We say that this round is successful if node v_i receives a message during this round. If the round is successful, then the waiting node in the next round is the next node v_{i+1} on the path (even if v_{i+1} is already active). Otherwise, node v_i remains the waiting node.

Let $t_0 = 0$, and for $i \geq 1$, let t_i be the round with the ith success, that is, the first round *after* round t_{i-1} when node v_i receives a message. Clearly $T_u \leq t_d$, since node u receives a message in round t_d but might have received a message earlier, along a path other than P. To show (12.3), we show that $t_d = \mathscr{O}(d + \log n)$ with probability at least $1 - 1/n^2$.

The outcome of the current round is not independent of the outcomes of the previous rounds, but (12.2) implies that the probability of success is always at least $1/4$. This implies easily that $t_d = \mathscr{O}(d \log n)$ with probability at least $1 - 1/n^2$. Indeed, for sufficiently large constant c,

$$\text{Prob}(t_d > cd \log n)$$

$$\leq \text{Prob}(c \log n \text{ succesive failures after the } i\text{th success, for some } 0 \leq i < d)$$

$$\leq \sum_{i=0}^{d-1} \text{Prob}(c \log n \text{ succesive failures after the } i\text{th success})$$

$$\leq n \left(\frac{3}{4} \right)^{c \log n} \leq \frac{1}{n^2}.$$

The above $\mathscr{O}(d \log n)$ bound on t_d gives only an $\mathscr{O}(D \log^2 n)$ bound on the running time of the algorithm RANDOMIZEDBROADCAST. To get a stronger bound on t_d, observe first that the expected value of t_d is at most $4d$ because (12.2) implies that the expected number of rounds we have to wait for the next success is at most 4. Thus, showing that $t_d = \mathscr{O}(d + \log n)$ with probability at least $1 - 1/n^2$ reduces to showing that the probability of a large deviation of t_d from its expected value is small.

A technical issue in this analysis is the fact that the trials in the sequence $\mathscr{A}$ are not independent. However, since the lower bound of $1/4$ on the probability of success in each trial in $\mathscr{A}$ holds for *every* pattern of outcomes in the previous trials, one can define a sequence of trials $\mathscr{B}$ with the following properties. The probability of success in each trial in $\mathscr{B}$ is equal to $p = 1/4$ *independently* of the outcomes of the previous trials (so $\mathscr{B}$ is a sequence of *binomial trials*), and the success in a trial t in $\mathscr{B}$ implies the success in trial t in $\mathscr{A}$. More precisely, let $q(a_1, a_2, \ldots, a_{t-1})$ denote the probability of success in trial t in $\mathscr{A}$ given a sequence $(a_1, a_2, \ldots, a_{t-1})$

of possible (positive probability) outcomes of the first $t-1$ trials, which is always at least p, and let $U_j \sim U(0,1)$, for $j \geq 1$, be independent uniform (continuous) random variables. For any sequence $(a_1, a_2, \ldots, a_t)$ of possible outcomes of the first t trials in $\mathscr{A}$, declare trial t in $\mathscr{B}$ a success if and only if a_t is the success and $U_t \leq p/q(a_1, a_2, \ldots, a_{t-1})$. Thus, given the outcomes $(a_1, a_2, \ldots, a_{t-1})$ of the first $t-1$ trials in $\mathscr{A}$ and the values of random variables U_j for $j \leq t-1$ (denote this condition by C), we know exactly the outcomes of the first $t-1$ trials in $\mathscr{B}$, while the conditional probability of the success in trial t in $\mathscr{B}$ is equal to:

$$\text{Prob}(\text{success in trial } t \text{ in } \mathscr{B} \mid C)$$
$$= \text{Prob}(\text{success in trial } t \text{ in } \mathscr{A} \text{ and } U_t \leq p/q(a_1, \ldots, a_{t-1}) \mid C)$$
$$= \text{Prob}(\text{success in trial } t \text{ in } \mathscr{A} \mid U_t \leq p/q(a_1, \ldots, a_{t-1}) \text{ and } C)$$
$$\times \text{Prob}(U_t \leq p/q(a_1, \ldots, a_{t-1}) \mid C)$$
$$= \text{Prob}(\text{success in trial } t \text{ in } \mathscr{A} \mid (a_1, \ldots, a_{t-1})) \times \text{Prob}(U_t \leq p/q(a_1, \ldots, a_{t-1}))$$
$$= q(a_1, \ldots, a_{t-1}) \cdot p/q(a_1, \ldots, a_{t-1}) \; = \; p.$$

This means that the probability of success in trial t in $\mathscr{B}$ is equal to p and does not depend on the outcomes of trials $1, 2, \ldots, t-1$.

Denoting by X_i the trial in $\mathscr{B}$ with the ith success, we have $X_i \geq t_i$, since whenever a trial in $\mathscr{B}$ is successful, the corresponding trial in $\mathscr{A}$ is also successful. The expected value of X_d is $4d$, and there are constants $\gamma > 4$ and $1 > \delta > 0$ such that for all $x \geq \gamma d$,

$$\text{Prob}(X_d > x) = \text{Prob}(\text{fewer than } d \text{ successes in the first } x \text{ trials in } \mathscr{B})$$
$$= \sum_{i=0}^{d-1} \binom{x}{i} p^i (1-p)^{x-i} \leq d \left(\frac{xe}{d}\right)^d \left(\frac{3}{4}\right)^x \leq 2^{-\delta x}. \tag{12.4}$$

The first inequality holds because $\binom{m}{k} \leq \left(\frac{me}{k}\right)^k$ for all $m \geq k \geq 1$. Now we have

$$\text{Prob}(t_d \geq (\gamma/\delta)(d + \log n)) \leq \text{Prob}(X_d \geq (\gamma/\delta)(d + \log n))$$
$$\leq 2^{-\gamma(d + \log n)} \leq \frac{1}{n^2}. \tag{12.5}$$

Thus $t_d = \mathcal{O}(d + \log n)$ with probability at least $1 - 1/n^2$, implying that the algorithm RANDOMIZEDBROADCAST completes broadcasting in $T = \mathcal{O}(D + \log n)$ rounds with probability at least $1 - 1/n$.

To bound the expected value of T, observe that (12.4) implies that for $x \geq \gamma d$, $\text{Prob}(T \geq x) \leq n 2^{-\delta x}$. Thus,

$$E(T) \leq (\gamma/\delta)(D + \log n) + \sum_{x \geq (\gamma/\delta)(D + \log n)} \text{Prob}(T \geq x)$$
$$\leq \mathcal{O}(D + \log n) + \sum_{x \geq (\gamma/\delta)(D + \log n)} n 2^{-\delta x} = \mathcal{O}(D + \log n).$$

12.3.4 Using Selective Families in Deterministic Distributed Broadcasting

In randomized distributed broadcasting in ad hoc radio networks, the selection of exactly one node from the $k \geq 1$ active neighbors of a node v is achieved by steps when each active node transmits with probability $1/\Theta(k)$. $\mathcal{O}(\log n)$ such steps ensure that node v receives a message from its neighbor with high probability. In deterministic distributed broadcasting, the selection of exactly one neighbor of v is based on the notion of *selective families of sets*. We illustrate how these combinatorial structures are used by describing the $\mathcal{O}(n \log^2 n)$-step deterministic broadcasting algorithm proposed in [33]. We call this algorithm DETERMINISTICBROADCAST.

We say that a family $\mathscr{R}$ of subsets of $\{1, 2, \ldots, n\}$ is *k-selective*, for a positive integer k, if for any nonempty set $Z \subseteq \{1, 2, \ldots, n\}$ with $k/2 \leq |Z| \leq k$, there is a set $R \in \mathscr{R}$ such that $|R \cap Z| = 1$. A *k-selective sequence* $\mathscr{S} = (S_1, S_2, \ldots, S_q)$ is an (arbitrary) linear order of the sets of a k-selective family. In a radio network, if the nodes in a set W transmit according to a k-selective sequence $\mathscr{S}$ (a node $w \in W$ transmits in step i, if and only if $w \in S_i$) while the nodes not in W remain silent, Z is the set of neighbors of a node v which are in W, and $k/2 \leq |Z| < k$, then for some step i, $|S_i \cap Z| = 1$ and node v receives a message. Note that the above definition of a k-selective family is slightly different than the definition used in the deterministic broadcasting algorithm in [33].

It can be shown by a simple probabilistic argument that for each positive $k \leq n$, there exists a k-selective family of size $\mathcal{O}(k \log n)$ [37]. An explicit construction of a k-selective family of size $\mathcal{O}(k \log^{\mathcal{O}(1)} n)$ was shown in [70]. Observe the large gap between the upper bounds on the number of steps required by the randomized and deterministic selections of (exactly) one node from an arbitrary group of k nodes: $\mathcal{O}(\log n)$ steps (with high probability) for the randomized selection but $\mathcal{O}(k \log n)$ steps for the deterministic selection. There is no hope for any significant improvement of the latter (deterministic) bound since a k-selective family is not only sufficient but also necessary in this context, and it was shown in [37] that the size of any k-selective family must be $\Omega(k \log(n/k))$.

In the RANDOMIZEDBROADCAST algorithm, the issue of potentially varied sizes of the active neighborhoods is dealt with by using different probabilities of transmission in different steps. These probabilities are $1/2^i$, for $i = 1, 2, \ldots, \lceil \log n \rceil$, and the steps when the active nodes transmit with probability $1/2^i$ take care of the active neighborhoods with sizes in $[2^{i-1}, 2^i]$. The DETERMINISTICBROADCAST algorithm uses 2^i-selective sequences $\mathscr{S}^{(i)}$ of size $\mathcal{O}(2^i \log n)$, for $i = 1, 2, \ldots, \lceil \log n \rceil$, applying them in an interleaved manner. The 2^i-selective sequence takes care of the active neighborhoods with sizes in $[2^{i-1}, 2^i]$.

The algorithm is a sequence of T *rounds*, and each round consists of $\lceil \log n \rceil$ steps; see Figure 12.2. The transmissions in steps i of each round are governed by the 2^i-selective sequence $\mathscr{S}^{(i)}$. More precisely, the transmissions in step i of round j, for $j = 1, 2, \ldots, |\mathscr{S}^{(i)}|$, are governed by the jth set in $\mathscr{S}^{(i)}$. Then a new repetition of $\mathscr{S}^{(i)}$ begins and the transmissions in step i of round $|\mathscr{S}^{(i)}| + j$, for $j = 1, 2, \ldots$,

$|\mathscr{S}^{(i)}|$, are governed by the jth set in $\mathscr{S}^{(i)}$, and so on. Sequence $\mathscr{S}^{(i)}$ is iterated in this way until the termination of the algorithm. If $S^{(i)}$ denotes the set in $\mathscr{S}^{(i)}$ which governs the transmissions in step i of the current round, then a node v transmits in this step if and only if $v \in S^{(i)}$ and v was active already at the beginning of the current repetition of sequence $\mathscr{S}^{(i)}$.

The analysis below shows that the bound T on the number of rounds can be set to be $\mathcal{O}(n\log n)$, implying an $\mathcal{O}(n\log^2 n)$ bound on the total number of steps. To simplify the analysis, we assume that for some integral constant c, $|\mathscr{S}^{(i)}| = c2^i\lceil\log n\rceil$, for each $i = 1, 2, \ldots, \lceil\log n\rceil$. Thus $|\mathscr{S}^{(i)}|$ is a multiplier of $|\mathscr{S}^{(j)}|$, for each $j < i$).

> **for** each node v in parallel **do**
>   **for** $i = 1$ to $\lceil\log n\rceil$ **do**
> $S^{(i)} =$ the next set in the 2^i-selective sequence $\mathscr{S}^{(i)}$
>   { if the end of $\mathscr{S}^{(i)}$ is reached, then repeat $\mathscr{S}^{(i)}$ from the beginning }
>   **if** v was active at the beginning of the current repetition of $\mathscr{S}^{(i)}$ **then**
> v transmits, if $v \in S^{(i)}$

Fig. 12.2 One round of algorithm DETERMINISTICBROADCAST

For a node $v \in V$, let $\mathscr{N}(v)$ denote the set of the neighbors of v, and let T_v be the round when v becomes active. The broadcasting is completed in $\max_{v\in V}\{T_v\}$ rounds. We take an arbitrary node v other than the source and show that $T_v = \mathcal{O}(n\log n)$. Let $P = (s = v_0, v_1, v_2, \ldots, v_d = v)$ be a shortest path from s to v. Let

$$L_d = \{v_d\},$$

$$L_i = \mathscr{N}(v_{i+1}) \setminus \bigcup_{j=i+1}^{d} L_j, \quad \text{for } i = d-1, d-2, \ldots, 0.$$

Figure 12.3 illustrates the definition of sets L_i. Observe that sets L_i are pairwise disjoint, $v_i \in L_i$, for each $0 \le i \le d$, and $\mathscr{N}(v_{i+1}) \subseteq \bigcup_{j=i}^{d} L_j$, for each $0 \le i \le d-1$.
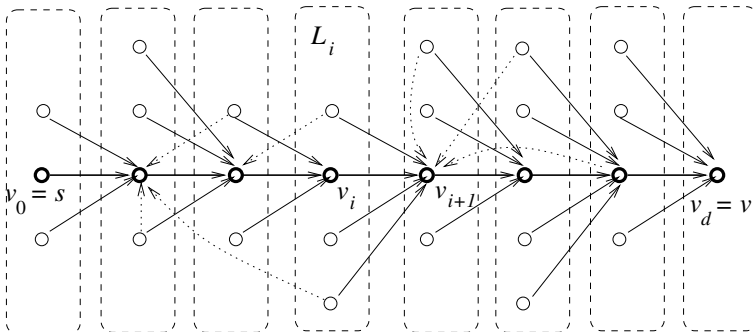
Fig. 12.3 A shortest path from s to v and sets L_i

Let t_i be the first round when a node in $\bigcup_{j=i}^{d} L_j$ becomes active. We have $t_0 = 0 < t_1 \le t_2 \le \cdots \le t_d = T_v$. We consider an arbitrary index i, $0 \le i \le d-1$, and show that $t_{i+1} - t_i = \mathcal{O}(|L_i| \log n)$. Assume that $t_{i+1} > t_i$. Thus, in round t_i a node in L_i becomes active but all nodes in $\bigcup_{j=i+1}^{d} L_j$ are still inactive. Let $t > t_i$ be the first round after round t_i when a new repetition of $\mathscr{S}^{(p)}$ starts, where $p = \lceil \log |L_i| \rceil$, and let $1 \le q \le p$ be such that the number of active nodes in L_i at the beginning of round t is in $[2^{q-1}, 2^q]$. Note that round t is also the first round of a new repetition of $\mathscr{S}^{(q)}$. Since a new repetition of $\mathscr{S}^{(p)}$ starts every $|\mathscr{S}^{(p)}| = c2^p \lceil \log n \rceil$ rounds, we have

$$t \le t_i + |\mathscr{S}^{(p)}| = t_i + c2^p \lceil \log n \rceil.$$

Since only the nodes in L_i which are active at the beginning of round t participate in the repetition of $\mathscr{S}^{(q)}$ which starts at this round, and $\mathscr{S}^{(q)}$ is a 2^q-selective sequence, there is a round t', $t \le t' < t + |\mathscr{S}^{(q)}|$, such that exactly one node w in L_i transmits in the qth step of this round. If $t_{i+1} \ge t'$, then no node in $\mathcal{N}(v_{i+1}) \setminus L_i \subseteq \bigcup_{j=i+1}^{d} L_j$ is active at the beginning of round t', so node $v_{i+1} \in \bigcup_{j=i+1}^{d} L_j$ receives a message in this round (from node w) and we must have $t_{i+1} = t'$. Thus, $t_{i+1} \le t'$, so

$$
\begin{aligned}
t_{i+1} \le t' \; &< \; t + |\mathscr{S}^{(q)}| \; \le \; t_i + c2^p \lceil \log n \rceil + c2^q \lceil \log n \rceil \; \le \; t_i + c2^{p+1} \lceil \log n \rceil \\
&\le \; t_i + 4c|L_i| \lceil \log n \rceil.
\end{aligned}
\tag{12.6}
$$

Inequality (12.6) holds for each $i = 0, 1, \ldots, d-1$, implying

$$T_v = t_d \le 4c \lceil \log n \rceil \sum_{i=0}^{d-1} |L_i| \le 4cn \lceil \log n \rceil = \mathcal{O}(n \log n),$$

where the last inequality follows from the fact that sets $|L_i|$ are pairwise disjoint.

12.4 Other Variants

12.4.1 Directed Graphs

A model based on *directed* graphs was used in [12, 18, 21, 22, 26, 28, 33, 37, 42, 45, 76]. The aim of these papers was to construct broadcasting algorithms working as fast as possible in arbitrary directed radio networks without knowing their topology. It turned out that in the directed setting, the complexity of broadcasting may depend on an additional parameter, namely, the maximum node degree Δ.

The randomized algorithms of [10, 42] apply also to directed networks; hence, broadcasting can be performed on directed networks in the harsh model by a randomized algorithm in time $\mathcal{O}(D \log(n/D) + \log^2 n)$, just as on undirected networks [42].

Deterministic broadcasting protocols in ad hoc radio networks were dealt with in [12, 13, 21, 22, 37, 46]. A scheme based on polynomials over finite fields was

presented in [21]. This scheme achieves broadcast in time $\mathcal{O}(D\frac{\Delta^2}{\log^2\Delta}\log^2 n)$, for arbitrary n-node networks with diameter D and maximum degree Δ. (The result was stated for undirected graphs, but it holds also for arbitrary directed graphs, defining D as the source radius, namely, the maximum distance from the source to any other node.) This direction was studied further in [13, 22]. A protocol completing broadcast in time $\mathcal{O}(D\Delta\log^{log\Delta} n)$ was constructed in [12]. Finally, an $\mathcal{O}(D\Delta\log^{\alpha} n)$-time protocol, for any $\alpha > 2$, was described in [37]. The protocol works for arbitrary directed graphs, and the exponent α can be decreased to 2 if the nodes are assumed to know n, and to 1 if the nodes know n and Δ. The above algorithms are efficient for networks with low diameter and node degrees. However, if D and Δ are large (say, linear in n), then the broadcasting time becomes $\Omega(n^2)$.

The first deterministic algorithm relying on universal selective families of sequences and avoiding the dependency on Δ, presented in [26], required $\mathcal{O}(n^{11/6})$ time. Successively faster algorithms were then developed in [28, 33, 45]. The fastest deterministic broadcasting algorithm currently available for directed networks in the harsh model has time $\mathcal{O}(n\log^2 D)$ [42]. The algorithm is oblivious and makes use of efficient deterministic selection sequences. On the other hand, a lower bound of $\Omega(n\log D)$ on deterministic broadcasting time was proved in [37] for directed networks. This lower bound holds also in a model allowing spontaneous wakeup (but still assuming collision-as-silence). See also [18, 26, 79].

It is worth noting that following the introduction of selective families in [26], a variety of probabilistic, combinatorial, and coding-based techniques were developed for constructing selective families and related structures such as strongly selective families, cover-free families, superimposed codes, and selectors, and these structures were used for broadcasting and other communication tasks in radio networks, such as wakeup and synchronization [28–30, 37–39, 42, 61, 70]. For a recent review of selection sequences and their uses, see [30].

12.4.2 Unit Disk Graphs

A more specific model of radio networks that has recently received considerable attention is based on assuming that the network nodes are placed in the two-dimensional plane and representing these nodes by their geometric positions in the plane. The underlying graph is thus no longer arbitrary. Rather, the transmission range of each node v is characterized as some region $R(v)$ around its location, and a node u can receive the transmissions of v if it belongs to $R(v)$. The regions are often assumed to be unit disks, implying that the transmission range of each node includes all nodes at distance at most 1 from it. In this case, the resulting network is a *unit disk graph* (UDG), where two nodes are joined by an edge if and only if their Euclidean distance is at most 1. Another common alternative is to allow a generalized representation as a (directed) disk graph, where radii of disks representing reachability regions may differ from node to node [43]. Reachability areas of arbitrary shapes were considered in [46, 82].

Broadcasting in such geometric radio networks and some of their variations was considered in [43, 46, 82, 88, 90].

Broadcasting schedules and bounds on $\hat{b}(n,D)$**.** In [90] it is proved that computing an optimal broadcasting schedule is NP-hard even when restricted to such graphs. That paper also gives an $\mathcal{O}(n\log n)$ algorithm for constructing an optimal broadcast schedule when the nodes are situated on a straight line. In [88] broadcasting was considered in networks with nodes randomly placed on a straight line. Fault-tolerant broadcasting in radio networks arising from regular locations of nodes on the line and in the plane, with reachability regions being squares and hexagons, rather than circles, is discussed in [82]. Broadcasting with restricted knowledge was considered in [46], but only the special case of nodes situated on the line was analyzed.

The problem of bounding $\hat{b}(n,D)$ becomes easier in the setting of UDG networks, and admits tight bounds of $\hat{b}(n,D) = \Theta(D)$. The lower bound is trivial, and the upper bound follows, e.g., from [65, 75], but can also be derived directly in a straightforward manner.

Distributed broadcasting algorithms in unknown topology. The first paper to study deterministic distributed broadcasting in arbitrary geometric radio networks with restricted knowledge of topology was [43]. Several models were studied assuming a positive knowledge radius, i.e., assuming that the knowledge available to a node concerns other nodes inside some disk around it. In the case of knowledge radius 0, an $\mathcal{O}(n)$ time broadcasting algorithm is shown for the spontaneous wakeup model, assuming that nodes are labeled by consecutive integers.

In the distributed context, where nodes are unaware of the network topology (including their immediate neighborhood), the efficiency of broadcasting turns out to depend not only on the diameter but also on one additional parameter, namely, the *spacing* among nodes [50]. Let d be a lower bound on the Euclidean distance between any two nodes of the network. Then, broadcasting time depends inversely on d, or in other words, it depends on the network *granularity*, $g = 1/d$. Assume each node of the network initially knows only its own coordinates in the Euclidean plane and the parameter d. Considering only deterministic broadcasting algorithms, the decisions made by a node at each round are based entirely on its coordinates and on the messages it received so far. An algorithm for broadcasting in the harsh model is presented in [50], completing broadcast in time $\mathcal{O}(Dg)$ in any UDG radio network of diameter D and granularity g. A matching lower bound is presented in [51]. It should be noted that in a network of diameter D and granularity g, n may be as large as $\mathcal{O}(D^2g^2)$; hence, $\mathcal{O}(Dg)$ is generally an improvement over the $\mathcal{O}(n)$ bound that follows from the algorithm of [43].

The problem was studied also for the spontaneous wakeup model (still assuming collision-as-silence). Two different broadcasting algorithms are given in [50], one working in time $\mathcal{O}(D+g^2)$ and the other in time $\mathcal{O}(D\log g)$. Depending on parameter values, one or the other of these algorithms may be more efficient. The combined algorithm obtained by interleaving these two algorithms completes broadcast in time $\mathcal{O}(\min\{D+g^2, D\log g\})$. A matching lower bound of $\Omega(\min\{D+g^2, D\log g\})$ on broadcasting time for this model is established as well. These results give a provable separation between the conditional and the spontaneous wakeup models for broad-

casting in UDG radio networks; for networks of small diameter (e.g., D bounded or polylogarithmic in g) the lower bound for the conditional wakeup model is significantly larger than the upper bound for the spontaneous wakeup model.

One may consider also a variant of the model based on restricted network configurations, where the nodes must be deployed on the points of a *d-spaced grid* in the plane. This variant has also been examined in [50, 51] within the harsh model. A lower bound of $\Omega(D\sqrt{g})$ on the time required by any broadcasting algorithm is established in [50], and an algorithm achieving broadcast in time $\mathcal{O}(Dg^{5/6}\log g)$ is presented in [51]. This implies a separation between the arbitrary deployment setting and the grid deployment setting in the harsh model, showing that broadcasting in the former setting is strictly harder.

The problem is studied in [75] in the spontaneous wakeup model with a more elaborate reception model (with different interference and transmission ranges, and a carrier sensing mechanism allowing some form of collision detection). The broadcast algorithm described therein is based on initially constructing a connected dominating set and subsequently a constant density spanner for the network (in expected time $\mathcal{O}(\Delta \log \Delta \log n + \log^4 n)$), and then using this spanner to disseminate the broadcast message with high probability in overall time $\mathcal{O}(D + \log n)$.

12.4.3 Related Work

This concluding section briefly reviews some of the results appearing in the literature on variations of the problem, the model, or the type of solution, that do not fall under the categories discussed so far.

As mentioned earlier, one may distinguish a special class of algorithms, referred to as *oblivious* distributed algorithms, where the actions taken by each node depend only on its identity and the time but not on the history of the execution. Oblivious algorithms for broadcasting are studied in [80], where it is shown that obliviousness degrades the time efficiency of broadcast. In particular, it is shown that there are oblivious randomized algorithms completing broadcast in $\mathcal{O}(n \min\{D, \log n\})$ time, and every such algorithm requires $\Omega(n)$ expected time for broadcasting. Matching bounds of $\Theta(n \min\{D, \sqrt{n}\})$ are given for broadcasting time by oblivious deterministic algorithms.

Much of the literature on broadcasting algorithms concerns the *static* case, where the network topology is fixed through time. In practice, there are applications where the network nodes are *mobile*, and their movements affect also the topology of the network, since a moving node exits the transmission range of some of its previous neighbors and enters the transmission range of some new nodes. For discussions of broadcasting techniques and protocols for mobile wireless ad hoc networks, see [92, 93].

The broadcast operation has been studied extensively in other (wired) types of communication networks, e.g., message passing networks (cf. [6, 7, 15, 55, 68, 87, 89]) and telephone networks (cf. [53, 54] and the surveys [57, 67]).

The literature reviewed herein considers broadcast as a single operation. In many cases, there is a need to perform a long sequence of message broadcasts repeatedly. Such settings were studied in [23], and later in [11], which presents a preprocessing stage such that subsequently, repeated broadcast operations can be performed at an average throughput of a broadcast operation every $\mathcal{O}(\log \Delta \log n)$ time slots, where Δ is the maximum degree of the network. Using a preconstructed spanner to speed up broadcasting was proposed in [5].

A related communication task that has been thoroughly studied in all types of communication networks is *gossiping*, where each of the nodes of the network (in parallel) must broadcast its local information to all the other nodes [20, 33, 41, 60, 62, 64, 66, 67, 85]. Also related is the task of *repetitious communication*, where each node repeatedly exchanges information with its neighbors [2, 56].

Yet another related type of distributed operation is the *wakeup* problem. In the context of radio networks, this problem was first studied in [61] for single-hop networks (modeled by complete graphs), and then in [27, 29, 32] for arbitrary networks. Randomized wakeup algorithms for radio networks were studied in [71]. In all these papers it was assumed that a subset of all nodes wake up spontaneously, possibly at different times, and have to wake up the other (initially dormant) nodes.

The model as defined herein assumes synchronous communication. The broadcasting problem has been studied also in the *asynchronous* model. In particular, upper and lower bounds for broadcasting time in asynchronous radio networks are established in [31] under a variety of possible adversarial models.

A number of papers study other complexity measures for broadcasting, such as the total number of transmissions [65] or the total energy requirements [3, 19, 35, 36, 52, 91]. These measures are sometimes studied in combination with time complexity [14, 44, 63].

In certain variants of broadcast, the problem definition includes some element of termination detection, e.g., they may require the source node to receive an acknowledgement informing it that the broadcasting process has completed. Issues of termination detection and acknowledgement delivery in radio networks are studied in [26, 59].

Fault-tolerant broadcasting in radio networks was considered in a number of papers [8, 9, 16, 17, 40, 73, 74, 82, 84, 86]. See [85] for a survey of earlier results concerning fault-tolerant protocols in communication networks, which includes models related to radio networks.

References

1. Alon, N., Bar-Noy, A., Linial, N., Peleg, D.: A lower bound for radio broadcast. J. Computer and System Sciences **43**, (1991), 290–298
2. Alon, N., Bar-Noy, A., Linial, N., Peleg, D.: Single round simulation on radio networks. J. Algorithms **13**, (1992), 188–210
3. Ambuhl, C., Clementi, A., Di Ianni, M., Lev-Tov, N., Monti, A., Peleg, D., Rossi, G., Silvestri, R.: Efficient algorithms for low-energy bounded-hop broadcast in ad-hoc wireless networks. In: Proc. 21st Symp. on Theoretical Aspects of Computer Science, LNCS 2996, 2004, 418–427
4. Awerbuch, B.: A new distributed depth-first-search algorithm. Information Processing Letters **20**, (1985), 147–150
5. Awerbuch, B., Baratz, A., Peleg, D.: Efficient broadcast and light-weight spanners. Technical Report CS92-22, The Weizmann Institute, Rehovot, Israel, 1992
6. Awerbuch, B., Cidon, I., Kutten, S., Mansour, Y., Peleg, D.: Optimal broadcast with partial knowledge. SIAM J. Computing **28**, (1998), 511–524
7. Awerbuch, B., Goldreich, O., Peleg, D., Vainish, R.: A tradeoff between information and communication in broadcast protocols. J. ACM **37** (1990), 238–256
8. Awerbuch, B., Even, S.: Reliable broadcast protocols in unreliable networks. Networks **16**, (1986), 381–396
9. Awerbuch, B., Segall, A.: A reliable broadcast protocol. IEEE Trans. Communications **31**, (1983), 896–901
10. Bar-Yehuda, R., Goldreich, O., Itai, A.: On the time complexity of broadcast in radio networks: an exponential gap between determinism and randomization. J. Computer and System Sciences **45**, (1992), 104–126
11. Bar-Yehuda, R., Israeli, A., Itai, A.: Multiple communication in multihop radio networks. SIAM J. Computing **22**, (1993), 875–887
12. Basagni, S., Bruschi, D., Chlamtac, I.: A mobility-transparent deterministic broadcast mechanism for ad hoc networks. IEEE/ACM Trans. Networking **7**, (1999), 799–807
13. Basagni, S., Myers, A. D., Syrotiuk, V. R.: Mobility-independent flooding for real-time multimedia applications in ad hoc networks. In: Proc. IEEE Emerging Technologies Symp. on Wireless Communications and Systems, 1999, 20.1–20.5
14. Berenbrink, P., Cooper, C., Hu, Z.: Energy efficient randomized communication in unknown ad hoc networks. In: Proc. 19th ACM Symp. on Parallel Algorithms and Architectures, 2007, 250–259
15. Bertsekas, D., Gallager, R.: Data Networks. Prantice Hall, New Jersey, 1987
16. Bhandari, V., Vaidya, N. H.: On reliable broadcast in a radio network. In: Proc. 24th ACM Symp. on Principles of Distributed Computing, 2005, 138–147
17. Bhandari, V., Vaidya, N. H.: Reliable broadcast in wireless networks with probabilistic failures. In: Proc. 26th Joint Conf. of IEEE Computer and Communication Societies (INFOCOM), 2007, 715–723
18. Bruschi, D., M. Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. Distributed Computing **10**, (1997), 129–135
19. G. Călinescu, Li, X. Y., Frieder, O., Wan, P. J.: Minimum-energy broadcast routing in static ad hoc wireless networks. In: Proc. 20th Joint Conf. of IEEE Computer and Communications Societies (INFOCOM), 2001, 1162–1171
20. Chandra, R., Ramasubramanian, V., Birman, K. P.: Anonymous gossip: improving multicast reliability in mobile ad-hoc networks. In: Proc. 21st Conf. on Distributed Computing Systems, 2001, 275–283
21. Chlamtac, I., A. Faragó. Making transmission schedule immune to topology changes in multihop packet radio networks. IEEE/ACM Trans. Networking **2**, (1994), 23–29
22. Chlamtac, I., Faragó, A., Zhang, H.: Time-spread multiple access (TSMA) protocols for multihop mobile radio networks. IEEE/ACM Trans. Networking **5**, (1997), 804–812

23. Chlamtac, I., Kutten, S.: On broadcasting in radio networks – problem analysis and protocol design. IEEE Trans. Communications **33**, (1985), 1240–1246
24. Chlamtac, I., Kutten, S.: Tree-based broadcasting in multihop radio networks, IEEE Trans. Computers **36**, (1987), 1209–1223
25. Chlamtac, I., Weinstein, O.: The wave expansion approach to broadcasting in multihop radio networks. IEEE Trans. Communications **39**, (1991), 426–433
26. Chlebus, B. S., L. Gąsieniec, Gibbons, A., Pelc, A., Rytter, W.: Deterministic broadcasting in unknown radio networks. Distributed Computing **15**, (2002), 27–38
27. Chlebus, B. S., Gasieniec, L., Kowalski, D. R., Radzik, T.: On the wake-up problem in radio networks. In: Proc. 32nd Int. Colloq. on Automata, Languages and Programming, LNCS 3580, 2005, 347–359
28. Chlebus, B. S., L. Gąsieniec, Ostlin, A., Robson, J. M.: Deterministic radio broadcasting. In: Proc. 27th Int. Colloq. on Automata, Languages and Programming 2000, LNCS 1853, 717–728
29. Chlebus, B. S., Kowalski, D. R.: A better wake-up in radio networks. In: Proc. 23rd ACN Symp. on Principles of Distributed Computing, 2004, 266–274
30. Chlebus, B. S., Kowalski, D. R.: Almost optimal explicit selectors. In: Proc. 15th Symp. on Fundamentals of Computation Theory, LNCS 3623, 2005, 270–280
31. Chlebus, B. S., Rokicki, M.: Asynchronous broadcast in radio networks, In: Proc. 11th Colloq. on Structural Information and Communication Complexity, 2004, LNCS 3104, 57–68
32. Chrobak, M., Gąsieniec, L., Kowalski, D. R.: The wake-up problem in multi-hop radio networks. In: Proc. 15th ACM-SIAM Symp. on Discrete Algorithms 2004, 985–993
33. Chrobak, M., Gąsieniec, L., Rytter, W.: Fast broadcasting and gossiping in radio networks. J. Algorithms **43**, (2002), 177-189
34. Cicalese, F., Manne, F., Xin, Q.: Faster centralized communication in radio networks. In: Proc. 17th Symp. on Algorithms and Computation, LNCS 4288, 2006, 339–348
35. Clementi, A. E. F., Huiban, G., Penna, P., Rossi, G., Verhoeven, Y. C.: Some recent theoretical advances and open questions on energy consumption in ad-hoc wireless networks. In: Proc. 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks, 23–38, 2002
36. Clementi, A. E. F., Crescenzi, P., Penna, P., Rossi, G., Vocca, P.: On the complexity of computing minimum energy consumption broadcast subgraphs. In: Proc. 18th Symp. on Theoretical Aspects of Computer Science, LNCS 2010, 121–131, 2001
37. Clementi, A. E. F., Monti, A., Silvestri, R.: Selective families, superimposed codes, and broadcasting on unknown radio networks. In: Proc. 12th Ann. ACM-SIAM Symp. on Discrete Algorithms, 2001, 709–718
38. Clementi, A. E. F., Crescenzi, P., Monti, A., Penna, P., Silvestri, R.: On computing ad-hoc selective families. In: Proc. 4th Workshop on Approximation Algorithms for Combinatorial Optimization Problems, LNCS 2129, 2001, 211–222
39. Clementi, A. E. F., Monti, A., Silvestri, R.: Distributed broadcast in radio networks of unknown topology. Theoretical Computer Science **302**, (2003), 337–364
40. Clementi, A. E. F., Monti, A., Silvestri, R.: Round robin is optimal for fault-tolerant broadcasting on wireless networks. J. Parallel and Distributed Computing **64**, (2004), 89–96
41. Czumaj, A., Gąsieniec, L., Pelc, A.: Time and cost trade-offs in gossiping. SIAM J. Discrete Mathematics **11**, (1998), 400–413
42. Czumaj, A., Rytter, W.: Broadcasting algorithms in radio networks with unknown topology. J. Algorithms **60**, (2006), 115–143
43. Dessmark, A., Pelc, A.: Broadcasting in geometric radio networks. J. Discrete Algorithms **5**, (2007), 187–201
44. Dessmark, A., Pelc, A.: Deterministic radio broadcasting at low cost. Networks **39**, (2002), 88–97
45. De Marco, G., Pelc, A.: Faster broadcasting in unknown radio networks. Information Processing Letters **79** (2001) 53–56
46. Diks, K., Kranakis, E., Krizanc, D., Pelc, A.: The impact of knowledge on broadcasting time in linear radio networks. Theoretical Computer Science **287**, (2002), 449–471

47. Elkin, M., Kortsarz, G.: Logarithmic inapproximability of the radio broadcast problem. J. Algorithms **52**(1), (2004), 8–25
48. Elkin, M., Kortsarz, G.: Polylogarithmic inapproximability of the radio broadcast problem. In: Proc. 7th Workshop on Approximation Algorithms for Combinatorial Optimization Problems, 2004, LNCS 3122, 105–116
49. Elkin, M., Kortsarz, G.: Improved schedule for radio broadcast. In: Proc. 16th ACM-SIAM Symp. on Discrete Algorithms, 2005, 222–231.
50. Emek, Y., L. Gąsieniec, Kantor, E., Pelc, A., Peleg, D., Su, C.: Broadcasting in UDG radio networks with unknown topology. In: Proc. ACM Symp. on Principles of Distributed Computing, 2007, 195–204
51. Emek, Y., Kantor, E., Peleg, D.: On the effect of the deployment setting on broadcasting in euclidean radio networks. In: Proc. ACM Symp. on Principles of Distributed Computing, 2008, 223–232
52. Ephremides, A., Nguyen, G. D., Wieselthier, J. E.: On the construction of energy-efficient broadcast and multicast trees in wireless networks. In: Proc. 19th Joint Conf. of IEEE Computer and Communications Societies (INFOCOM), 585–594, 2000
53. Farley, A. M.: Minimal broadcast networks. Networks **9**, (1979), 313–332
54. Farley, A. M., Hedetniemi, S. T., Mitchell, S., Proskurowski, A.: Minimum broadcast graphs. Discrete Mathematics **25**, (1979), 189–193
55. Feige, U., Peleg, D., Raghavan, P., Upfal, E.: The complexity of randomized broadcast. J. on Random Structures & Algorithms **1**, (1990), 447–460
56. Fernandez, A., Mosteiro, M. A., Thraves, C.: Deterministic communication in the weak sensor model. In: Proc. 11th Conf. On Principles Of Distributed Systems, LNCS 4878, 2007, 119–131
57. Fraigniaud, P., Lazard, E.: Methods and problems of communication in usual networks. Discrete Applied Mathematics **53**, (1994), 79–133
58. Gaber, I., Mansour, Y.: Centralized broadcast in multihop radio networks. J. Algorithms **46**, (2003), 1–20
59. Gargano, L., Pelc, A., Pérennes, S., Vaccaro, U., Efficient communication in unknown networks, Networks **38**, (2001), 39–49
60. Gąsieniec, L., Pagourtzis, A., Potapov, I., Radzik, T.: Deterministic communication in radio networks with large labels. Algorithmica **47**, (2007), 97–117
61. Gąsieniec, L., Pelc, A., Peleg, D.: The wakeup problem in synchronous broadcast systems. SIAM J. Discrete Mathematics **14**, (2001), 207–222
62. Gąsieniec, L., Peleg, D., Xin, Q.: Faster communication in known topology radio networks. In: Proc. 24th ACM Symp. on Principles Of Distributed Computing, 2005, 129–137
63. Gąsieniec, L., Kantor, E., Kowalski, D. R., Peleg, D., Su, C.: Energy and time efficient broadcasting in known topology radio networks. In: Proc. 21st Symp. on Distributed Computing, 2007, LNCS 4731, 253–267
64. Gąsieniec, L., Radzik, T., Xin, Q.: Faster deterministic gossiping in directed ad hoc radio networks. In: Proc. 9th Scandinavian Workshop on Algorithm Theory, LNCS 3111, 2004, 397–407
65. Gandhi, R., Parthasarathy, S., Mishra, A.: Minimizing broadcast latency and redundancy in ad hoc networks. In: Proc. 4th ACM Symp. on Mobile ad hoc networking and computing, 2003, 222–232.
66. Haas, Z. J., Halpern, J. Y., Li, L.: Gossip-based ad hoc routing. IEEE/ACM Trans. Networking **14**, (2006), 479–491
67. Hedetniemi, S., Hedetniemi, S., Liestman, A.: A survey of gossiping and broadcasting in communication networks. Networks **18**, (1988), 319–349
68. Humblet, P. A., Soloway, S. R.: Topology broadcast algorithms. Computer Networks **16**, (1989), 179–186
69. Hwang, F. K.: The time complexity of deterministic broadcast radio networks. Discrete Applied Mathematics **60**, (1995), 219–222
70. Indyk, P.: Explicit constructions of selectors and related combinatorial structures, with applications. In: Proc. 13th ACM-SIAM Symp. on Discrete Algorithms, 2002, 697–704

71. Jurdzinski, T., Stachowiak, G.: Probabilistic algorithms for the wakeup problem in single-hop radio networks. In: Proc. 13th Int. Symp. on Algorithms and Computation (ISAAC 2002), LNCS 2518, 535 – 549

72. Kesselman, A., Kowalski, D. R.: Fast distributed algorithm for convergecast in ad hoc geometric radio networks. In: Proc. 2nd Conf. on Wireless on Demand Network Systems and Service, 2005, 119–124

73. C.-Koo, Y.: Broadcast in radio networks tolerating byzantine adversarial behavior. In: Proc. 23rd ACM Symp. on Principles of Distributed Computing, 2004, 275–282

74. C.-Koo, Y., Bhandari, V., Katz, J., Vaidya, N. H.: Reliable broadcast in radio networks: the bounded collision case. In: Proc. 25th ACM Symp. on Principles of Distributed Computing, 2006, 258–264

75. Kothapalli, K., Onus, M., Richa, A., Scheideler, C.: Efficient broadcasting and gathering in wireless ad hoc networks. In: Proc. IEEE Symp. on Parallel Architectures, Algorithms and Networks, 2005, 346–351

76. Kowalski, D. R., Pelc, A.: Faster deterministic broadcasting in ad hoc radio networks. SIAM J. Discrete Mathematics **18**, (2004), 332–346

77. Kowalski, D. R., Pelc, A.: Centralized deterministic broadcasting in undirected multi-hop radio networks. In: Proc. 7th Workshop on Approximation Algorithms for Combinatorial Optimization Problems, 2004, LNCS 3122, 171–182

78. Kowalski, D. R., Pelc, A.: Time of deterministic broadcasting in radio networks with local knowledge, SIAM J. Computing **33**, (2004), 870–891

79. Kowalski, D. R., Pelc, A.: Broadcasting in undirected ad hoc radio networks. Distributed Computing **18**, (2005), 43–57

80. Kowalski, D. R., Pelc, A.: Time complexity of radio broadcasting: adaptiveness vs. obliviousness and randomization vs. determinism. Theoretical Computer Science **333**, (2005), 355–371

81. Kowalski, D. R., Pelc, A.: Optimal deterministic broadcasting in known topology radio networks. Distributed Computing **19**, (2007), 185–195

82. Kranakis, E., Krizanc, D., Pelc, A.: Fault-tolerant broadcasting in radio networks. J. Algorithms **39**, (2001), 47–67

83. Kushilevitz, E., Mansour, Y.: An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks. SIAM J. Computing **27**, (1998), 702–712

84. Pagani, E., Rossi, G. P.: Reliable broadcast in mobile multihop radio networks. In: Proc. 3rd ACM/IEEE Conf. on Mobile Computing and Networking, 1997, 34–42

85. Pelc, A.: Fault-tolerant broadcasting and gossiping in communication networks. Networks **28**, (1996), 143–156

86. Pelc, A., Peleg, D.: Broadcasting with locally bounded Byzantine faults. Information Processing Letters **93**, (2005), 109–115

87. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM, 2000

88. Ravishankar, K., Singh, S.: Broadcasting on $[0, L]$. Discrete Applied Mathematics **53**, (1994), 299–319

89. Segall, A.: Distributed network protocols. IEEE Trans. Information Theory **29**, (1983), 23–35

90. Sen, A., Huson, M. L.: A new model for scheduling packet radio networks. In: Proc. 15th Joint Conf. of IEEE Computer and Communication Societies (INFOCOM), 1996, 1116–1124

91. Song, W.-Z., Li, X.-Y., Frieder, O., Wang, W.: Localized topology control for unicast and broadcast in wireless ad hoc networks. IEEE Trans. Parallel & Distributed Systems **17**, (2006), 321–334

92. Toh, C. K.: Ad Hoc Mobile Wireless Networks. Prantice Hall, 2002

93. Williams, B., Camp, T.: Comparison of broadcasting techniques for mobile ad hoc networks. In: Proc. 3rd ACM Symp. on Mobile Ad Hoc Networking and Computing, 2002, 194–205

Chapter 13
Energy Consumption Minimization in Ad Hoc Wireless and Multi-interface Networks

Alfredo Navarra, Ioannis Caragiannis, Michele Flammini, Christos Kaklamanis, and Ralf Klasing

Abstract This chapter deals with energy consumption issues in wireless networks. In such networks, energy is a scarce resource and, hence, it must be used efficiently. Under these circumstances, we consider two interesting combinatorial optimization problems: *Minimum Energy Broadcast Routing* and *Cost Minimization in Multi-interface Networks*. The goal of the first problem is to perform broadcasting from a given source while minimizing the overall energy required for communication. The second problem refers to the choice of activating a set of available communication interfaces at the network nodes in order to satisfy the required connections in a wireless multi-interface network with minimum total cost. While Minimum Energy Broadcast Routing has been studied extensively during recent years, Cost Minimization in Multi-interface Networks is rather new. For both problems we survey recent complexity results and approximation algorithms under different assumptions.

Key words: energy saving, wireless networks, multi-interface networks, broadcasting, approximation algorithms

Alfredo Navarra
Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Via Vanvitelli 1, 06123 Perugia, Italy, e-mail: `navarra@dmi.unipg.it`

Ioannis Caragiannis · Christos Kaklamanis
Computer Technology Institute & Department of Computer Engineering and Informatics, University of Patras, 26500 Rio, Greece, e-mail: `caragian, kakl@ceid.upatras.gr`

Michele Flammini
Dipartimento di Informatica, Università degli Studi dell'Aquila, Via Vetoio, Loc. Coppito, 67100 L'Aquila, Italy, e-mail: `flammini@di.univaq.it`

Ralf Klasing
CNRS - LaBRI - Université Bordeaux 1, France, e-mail: `klasing@labri.fr`

13.1 Introduction

In recent years *wireless networks* have been widely deployed, mostly because of the recent drop in equipment prices and due to the features provided by the new technologies. Unlike traditional *wired networks* where signals pass from one device to another through physical cables, in wireless networks data transmissions from each node (station) occur in the open air and within a given coverage area. In this scenario, considerable attention has been devoted to the so-called *ad hoc* wireless networks, due to their potential applications in emergency disaster relief, on the battlefield, in impervious areas, and so on [29, 44]. Ad hoc wireless networks do not require any fixed infrastructure. The network is simply a collection of devices that can communicate with each other according to proximity and available common protocols and interfaces.

In this chapter we consider two important problems arising in the context of ad hoc wireless networks: *Minimum Energy Broadcast Routing* and *Cost Minimization in Multi-interface Networks*. The first problem considers the need of broadcasting information from a given source to all other network nodes when the network nodes are equipped with omnidirectional antennas. The second problem aims at establishing connections among heterogeneous nodes equipped with a set of interfaces that can be activated at a different cost.

The common objective in both the above scenarios is to minimize the total energy consumption in order to keep the network alive as long as possible. Energy is in fact a scarce resource in wireless ad hoc networks, and communication strongly depends on it.

The chapter is organized as follows. Section 13.2 is devoted to the Minimum Energy Broadcast Routing problem. First, some motivation for the problem is provided and, then, it is formally defined. Several results are surveyed and details are given for interesting analysis techniques. Section 13.3 is devoted to Cost Minimization in Multi-interface Networks. Again, the problem is first motivated, then formally defined, and a list of results is presented emphasizing some interesting techniques. Finally, Section 13.4 provides conclusive remarks and interesting directions of future research.

13.2 Minimum Energy Broadcast Routing

The study of a basic communication pattern such as broadcasting is of main interest in the context of wireless ad hoc networks. Broadcasting can in fact be used to set up the network or to rapidly spread useful information. The wireless environment allows all devices in the range of a transmitter x to receive messages sent by x. The range of transmissions basically depends on the environment where devices are distributed. According to the widely used power attenuation model [42], when a station s transmits with power P_s, a station r can receive messages from s if and only if $P_s > \beta dist(s,r)^\alpha$, where $dist(s,r)$ is the Euclidean distance between s and

r, α is a parameter which depends on the environment with typical values between 2 and 6, and β is a positive parameter known as the *reception quality threshold*. For the sake of simplicity, from now on we normalize parameter β to 1. Due to the nonlinearity of power attenuation, multi-hop transmission of messages through intermediate devices may result in energy conservation. The main property of wireless ad hoc networks is usually the lack of a fixed infrastructure for routing purposes. A natural issue arising in this setting is that of supporting broadcasting with minimum total energy consumption. This problem, called *Minimum Energy Broadcast Routing* (MEBR), is well-known in the literature and it has been extensively studied (see [1, 2, 5, 7–9, 12, 13, 16, 19, 20, 24–26, 31, 33, 36, 41]).

13.2.1 Definitions and Notation

Given a set of stations S, let $G(S)$ be the complete weighted directed graph whose nodes are the stations in S and in which the weight $w(x,y)$ of each edge (x,y) is the power required at x in order to transmit correctly to node y. A power assignment for S is a function $p : S \rightarrow \mathbb{R}_+$ assigning a transmission power $p(x)$ to every station x in S. A power assignment p for S yields a directed communication graph $G^p = (S, A)$ such that, for each directed edge (x,y) of $G(S)$, (x,y) belongs to A if and only if $p(x) \geq w(x,y)$, i.e., if x can correctly transmit to y. In this case, we say that y is within the range of x. The total cost of a power assignment p is then

$$cost(p) = \sum_{x \in S} p(x).$$

MEBR takes as input $G(S)$ together with a source station $s \in S$ and consists of finding a power assignment p of minimum cost such that G^p contains a directed spanning tree rooted at s (and directed towards the leaves). We call such a power assignment an *optimal power assignment* and denote its cost by $m^*(S,s)$.

The weight function $w : E \mapsto \mathbb{R}_+$ is usually symmetric (i.e., $w(x,y) = w(y,x)$ for each pair of stations x, $y \in S$). Nonsymmetric weight functions can be used to capture the irregularity of the environment or situations where stations use batteries of different types which may operate on different fixed energy levels. An important case of symmetric weight functions arises in the geometric version of MEBR. In this case, the stations of S correspond to points in a $d-$dimensional Euclidean space and the weight function is defined as $w(x,y) = dist(x,y)^\alpha$, where $dist$ is the Euclidean distance and $\alpha \geq 1$ is a positive parameter. Equivalently, in this case, we seek a range assignment $r : S \rightarrow \mathbb{R}_+$ such that the range $r(x)$ of a station x denotes the maximum Euclidean distance from x at which signals can be correctly received. Again, a range assignment r for S yields a directed communication graph $G^r = (S, A)$ such that the directed edge (x,y) belongs to A if and only if y is at distance at most $r(x)$ from x. We use the notation $G_\alpha(S)$ and $m^*_\alpha(S,s)$ to denote the input graph and the cost of the optimal range assignment in the geometric case.

13.2.2 The Geometric Version of MEBR

We first study the geometric version of MEBR. In general, the geometric version of MEBR is NP-hard, while it is solvable in polynomial time when $\alpha = 1$ or $d = 1$ [13, 20]. Many heuristics and corresponding experimental results can be found in the literature. We can find the Shortest Paths Tree (SPT), the Minimum Spanning Tree (MST), and the Broadcast Incremental Power (BIP) in [42]; the Iterative Maximum-Branch Minimization (IMBM) in [37]; the Adaptive Broadcast Consumption (ABC) in [33]; a refined BIP version in [43]; and many Integer Linear Programming approaches like the ones in [22, 30, 33, 43] (see also [2] for a comparative experimental study).

While such heuristics have been observed experimentally to perform pretty well on random instances of MEBR, the only one for which extended analytical studies were done is the MST heuristic. It is based on the idea of tuning ranges so that the communication graph contains a minimum spanning tree (see Section 1.5.1.1) of the cost graph $G(S)$. More precisely, denote by $T(S)$ a minimum spanning tree of $G(S)$. The MST heuristic considers $T(S)$ rooted at the source station s, directs the edges of $T(S)$ towards the leaves, and sets the power $p(x)$ of every internal station x of $T(S)$ with $k > 0$ children $x_1, \ldots, x_k$ in such a way that $p(x) = max_{i=1,\ldots,k} w(x, x_i)$. In other words, p is the power assignment of minimum cost inducing the directed tree derived from $T(S)$, and is such that $cost(p) \leq c(T(S))$, where $c(T(S))$ denotes the total cost of the edges in $T(S)$. Therefore, the approximation ratio of the heuristic is bounded by the ratio between the cost of a minimum spanning tree of $G(S)$ and the optimal power cost $m^*(S, s)$.
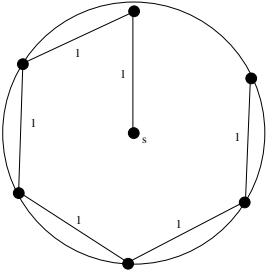
Fig. 13.1 The lower bound of 6 for the MST heuristic. On this instance, there is a source station s and six additional stations, five of which are on the circumference of a circle of radius $1 + \varepsilon$ centered at s. The MST heuristic produces a tree (path) consisting of six edges of total cost 6. The optimal solution is a star connecting s to the other six nodes at a cost of $1 + \varepsilon$. The approximation ratio can become arbitrarily close to 6 by selecting ε to be arbitrarily small

For the two-dimensional case (which is actually the case that has been given most attention in the literature), a lower bound of 6 on the approximation ratio of the MST heuristic was provided in [42] (see Figure 13.1). The first constant upper bound was provided in [19]. The analysis led to an approximation ratio of 40 for

MST, immediately reduced to 20 by the same authors. The general idea behind the analysis is to represent each edge of $G(S)$ chosen by the MST heuristic by a two-dimensional shape and evaluate the area occupied by all such shapes, since the cost of a solution provided by the MST heuristic (for $\alpha = 2$) is proportional to the considered area. The 40-approximation was obtained by associating with each edge e a circle whose diameter is the length of e (see Figure 13.2.a). The 20-approximation arises by associating a circle with each edge e whose center is the middle point of e and diameter is half the length of e (see Figure 13.2.b). A further improvement obtained by using the same technique but varying the shape was given in [33, 41]. The obtained approximation ratio is 12.15 and the shape associated with each edge e is a rhombus whose bigger diagonal coincides with e, and the angles at its endpoints formed by the sides of the rhombus are 60 degrees (see Figure 13.2.c). Note that the third shape implies an interesting property for which no overlap occurs among two shapes associated with two different edges of the minimum spanning tree. By refining the geometrical arguments but without changing the rhombus shape, a better bound of 10.86 was obtained in [8].
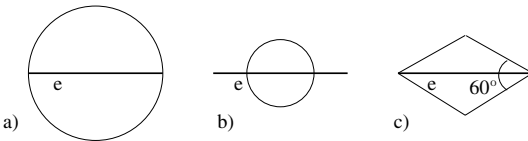
Fig. 13.2 The three shapes associated with an edge e of a spanning tree. The choice of shape a) leads to a 40-approximation, b) to 20, and c) to 12.15.

A different method was used in [24, 31]. A new process to evaluate the approximation factor of the MST heuristic was introduced, leading first to an upper bound of 8 and then to 6.33 by more refined geometrical arguments. The basic idea, which will be explained in detail in the next section, is to grow one circle centered at each node of the network until all the circles belong to the same connected component, that is, the union of all the circles forms one and only one delimited area. The bounds of the MST heuristic were then determined by evaluating the covered area by such a process. This method was also extended to the more general d-dimensional case for which a $(3^d - 1)$-approximation ratio has been obtained. It is worth mentioning that, for any Euclidean dimension d and power $\alpha \geq d$, the MST heuristic is lower-bounded by the so-called d-dimensional *kissing number* [19]. More precisely, the d-dimensional kissing number is the maximum number of mutually disjoint d-spheres (or hyperspheres) of a given radius r that can simultaneously touch a d-sphere of the same radius r in the d-dimensional Euclidean space [21]. For the three-dimensional Euclidean space, for instance, the kissing number is 12 while the upper bound provided by [24, 31] is 26. Such a bound has been improved in [38] to 18.8 by extending to the three-dimensional space arguments similar to the ones that led the upper bound for the two-dimensional case from 8 to 6.33. In [1] the gap between the lower bound of 6 and the upper bound of 6.33 for the MST heuristic

in the two-dimensional case was finally closed by decreasing the upper bound to 6. The author provided a new analysis technique based on the Delaunay triangulation.

An interesting direction for providing worst-case scenarios and studying the performance of the MST or other heuristics was given in [26]. By means of a randomized procedure, the approach showed an almost tight 4-approximation ratio of the MST heuristic in the case of uniform high-density distributions of the radio stations.

13.2.3 An 8-Approximation Upper Bound for the MST Heuristic

In this section, we present the main ideas that can be used in the analysis of the MST heuristic. The particular arguments used in the following yield an upper bound of 8 on the approximation ratio and are relatively easy to follow. The improved results in [31] and [1] follow significantly more involved analysis.

Given a graph G and a weight function w defined on its edges, for any $c \in \mathbb{R}_+$, let $N(G,c)$ be the number of connected components in the graph obtained from G by keeping only the edges $e \in E$ such that $w(e) \leq c$. Then, the cost $MST(G)$ of a minimum spanning tree for G is given by the following lemma.

Lemma 13.1 (Frieze and McDiarmid [27]). $MST(G) = \int_0^\infty (N(G,c) - 1)dc$.

For any subset of stations $Q \subseteq S$, let $G_\alpha(Q)$ be the subgraph of $G_\alpha(S)$ induced by Q. Also, let $n(Q,r) = N(G_\alpha(Q), r^\alpha)$, that is, the number of connected components in $G_\alpha(Q)$ obtained by maintaining only the edges between the nodes at distance at most r in Q. Recalling that each edge (x,y) has cost $w(x,y) = dist(x,y)^\alpha$ and exploiting Lemma 13.1 by substituting the variable c with r^α, we obtain the following corollary.

Corollary 13.1 (Klasing et al. [31]). *For any subset of stations $Q \subseteq S$, $MST(G_\alpha(Q))$* $= \alpha \int_0^\infty (n(Q,r) - 1)r^{\alpha-1}dr$.

Now, the main argument in the proof is the following. We address the case $\alpha = 2$ since the upper bound for $\alpha \geq 2$ can be directly inferred [24, 31]. Consider an optimal power assignment of cost $m^*(S,s)$ in which k stations $x_1, ..., x_k$ are assigned nonzero power. For $i = 1, ..., k$, denote by r_i the range of station x_i and let Q_i be the set of stations within the range of station x_i. We will show that $MST(G_\alpha(Q_i)) \leq 8r_i^\alpha$. In this way, we will obtain that

$$MST(G_\alpha(S)) \leq \sum_{i=1}^k MST(G_\alpha(Q_i)) \leq 8\sum_{i=1}^k r_i^\alpha = 8m^*(S,s),$$

thus proving the upper bound. Without loss of generality, we consider a set of stations $Q \subseteq S$ such that there exists a station $x \in Q$ with $\max_{y \in Q} dist(x,y) = 1$. Thus, all the points of Q belong to a circle of radius 1 in the plane, from now on denoted by C_1, and the cost of each edge of the weighted complete graph representing the input network is proportional to the square of the distance between its endpoints.

For simplicity of notation, for any set of stations Q, $G_2(Q)$ is denoted simply as $G(Q)$.

Let $G(Q,r)$ be the graph obtained by considering only the edges of $G(Q)$ of length at most r, or equivalently of cost at most r^2, and let $CC(Q,r)$ be the set of the connected components of $G(Q,r)$. Let r_{max} be the minimum r such that $G(Q,r)$ is connected (i.e., $n(Q,r_{max}) = |CC(Q,r_{max})| = 1$). Then, directly from Corollary 13.1,

$$MST(G) = 2 \int_0^{r_{max}} (n(Q,r) - 1) r \, dr.$$

For the sake of readability, from now Q is dropped from the notation, so that G, $G(r)$ $CC(r)$, $n(r)$, and r_{max} will denote $G(Q)$, $G(Q,r)$ $CC(Q,r)$, $n(Q,r)$ and $r_{max}(Q)$, respectively.

Theorem 13.1 ([31]). *Given any subset of stations $Q \subseteq S$ within a circle of radius 1, $MST(G(Q)) \leq 8$.*

$$|CC(0)| = 7, \quad r = 0$$

$$|CC(r_1)| = 6, \quad r = \frac{r_1}{2}$$

$$|CC(r_2)| = 5, \quad r = \frac{r_2}{2}$$

$$|CC(r_{max})| = 1, \quad r = \frac{r_{max}}{2}$$
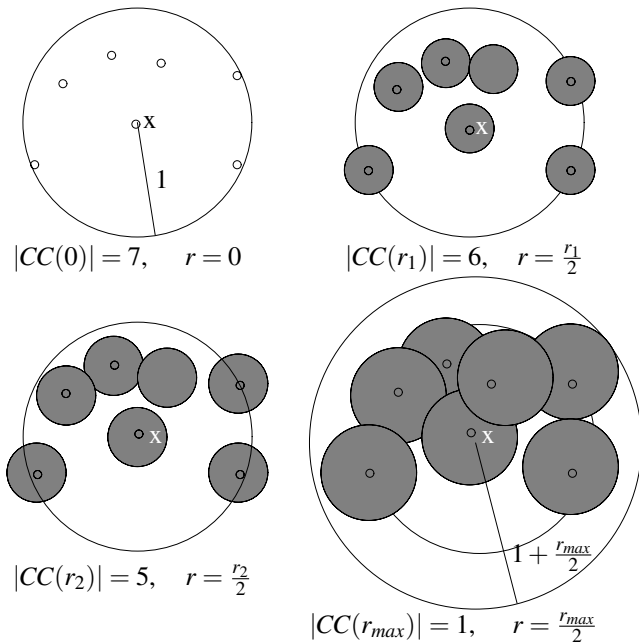
Fig. 13.3 The expanding process described in Theorem 13.1

The proof of Theorem 13.1 considers a growing process in which circles of equal radii centered at the stations of Q are synchronously grown starting from a radius $r = 0$ till $r = \frac{r_{max}}{2} \leq \frac{1}{2}$; i.e., the process ends when $G(2r)$ becomes connected. This is accomplished by increasing at any infinitesimal step the current radii, all equal to a given r, by dr.

For a set of stations P and a radius r, let $P(r)$ be the set of the points in the plane contained in the union of all the circles of radius r associated with the stations of P. Let $a(P,r)$ be the area of $P(r)$.

Suitable lower and upper bounds on the total area $a(Q, \frac{r_{max}}{2})$ covered by all the circles related to Q at the end of the process, that is, when all the radii are equal to $\frac{r_{max}}{2}$, can be determined as follows.

Consider a connected component $P \in CC(2r)$ of $G(2r)$. Then, since two circles of radius r centered in two stations at distance at most $2r$ touch each other, $P(r)$ corresponds to a closed region of the plane having perimeter $p(P,r)$ equal to at least $2\pi r$, that is, equal to at least the perimeter of a single circle of radius r centered in a station of P. Thus, the increase $p(P,r)dr$ of $a(P,r)$ when r is increased by an infinitesimal step dr is at least $2\pi r dr$.

If $P \in CC(2r)$ and $R \in CC(2r)$ are two different connected components of $G(2r)$, $P(r) \cap R(r) = \emptyset$, as any point belonging to the intersection would contradict the fact that the distance between any station in P and any station in R is strictly greater than $2r$. Therefore,

$$
\begin{aligned}
a(Q, \frac{r_{max}}{2}) &= \int_0^{\frac{r_{max}}{2}} \sum_{P \in CC(2r)} p(P,r)dr \\
&\geq \int_0^{\frac{r_{max}}{2}} n(2r)2\pi r dr \\
&= \frac{1}{4}2\pi \int_0^{r_{max}} n(r)r dr \\
&= \frac{1}{4}2\pi \int_0^{r_{max}} (n(r)-1)r dr + \frac{1}{4}2\pi \int_0^{r_{max}} r dr \\
&= \frac{\pi}{4}MST(G) + \frac{\pi}{4}r_{max}^2.
\end{aligned}
$$

Moreover, the total region of the plane covered by the union of all the circles related to Q of radius $\frac{r_{max}}{2}$, that is, $Q(\frac{r_{max}}{2})$, is included in a circle of radius $1 + \frac{r_{max}}{2}$ centered at the station x such that $dist(x,y) \leq 1$ for every $y \in Q$ (see Figure 13.3), so that $a(Q, \frac{r_{max}}{2}) \leq \pi + (\frac{r_{max}}{2})^2$. Therefore,

$$
\frac{\pi}{4}MST(G) + \frac{\pi}{4}r_{max}^2 \leq a(Q, \frac{r_{max}}{2}) \leq \pi(1 + \frac{r_{max}}{2})^2
$$

and thus, since $r_{max} \leq 1$,

$$
MST(G) \leq 4(1 + \frac{r_{max}}{2})^2 - r_{max}^2 = 4(1 + r_{max}) \leq 8.
$$

13.2.4 Experimental Studies with the MST Heuristic

We now show an interesting technique from [26] for obtaining "bad" instances for the MST heuristic. The goal is to maximize the cost of a possible MST inside C_1 considering its center s as the source. This was done in order to better understand the actual quality of the performance of the MST heuristic over interesting instances that are more representative of real-world applications. Starting from random instances, the maximization consists of slight movements of the nodes according to some useful properties of the MST construction. For instance, if we want to increase the cost of an edge of the MST, the easiest idea is to increase the distance of its endpoints. Let us now consider a node $v \neq s$ of a generic instance given as input. We consider the degree of such a node in the undirected tree obtained from the MST heuristic before assigning the directions. Let $N(v) = \{v_1, v_2, \ldots, v_k\}$ be the set of the neighbors of v in such a tree. We evaluate the median point $p = (x, y)$, whose coordinates are given by the average of the corresponding coordinates of the nodes in $N(v)$. The idea is then to move the node v farther from p but, of course, inside the considered circle. In general this should augment the cost of the MST on the edge connecting the node v to the rest of the tree (see Figure 13.4).
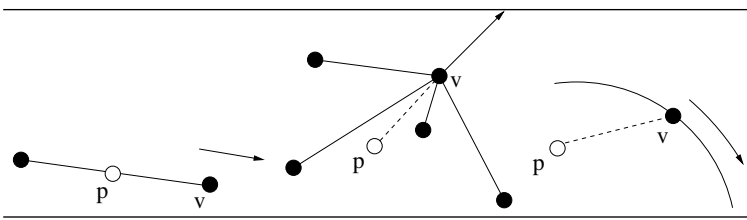
Fig. 13.4 Augmenting the edge costs when a node has one or more neighbors and when it is on the circumference of C_1

It can also happen that such a movement completely changes the structure of the MST, reducing the initial cost. In that case we do not validate the movement. Given an instance, the augmenting algorithm performs this computation for each node twisting over all the nodes but s, until no movements are allowed. Therefore, in order to give to a node a "second chance" to move, we can repeat such computations for a fixed number of rounds. Note that when a node reaches the border that is the circumference of C_1, the only allowed movement is over such a circumference.

Another way to increase the cost of the MST is to try to delete a node. The chosen candidate is the node with the highest degree. The idea behind this choice is that the highest degree node could be considered as the intermediary node to connect its neighbors, so removing it, a "big hole" will probably appear. On the one hand, this means that the distances to connect the remaining disjoint subtrees should increase the overall cost. On the other hand, we are creating more space for further movements. After a deletion, the algorithm starts again with the movements. Indeed, the deletion can be considered as a movement in which two nodes coincide. If the

deletion does not increase the cost of the current MST, it is not validated. In such a
case, the next step will be the deletion of the second highest degree node and so on.
The whole procedure is repeated until no movements and no deletions are allowed.
Note that eventually the algorithm can be repeated several times in order to obtain
more accurate results. Sometimes, in fact, it can happen that the algorithm is stuck
in some local maximum. Due to its randomness on the movements, the more it is
executed, the higher is the probability to exit from such a situation.

The algorithm was evaluated over hundreds of instances from five up to 100
nodes. Table 13.1 shows the average and the maximum costs obtained on random
instances using and not using the augmenting method (ε represents the maximum
distance allowed for movements).

Table 13.1 The average and the maximum costs obtained on standard random instances and using
the previous augmenting algorithm on instances of five up to 100 nodes and ε equal to 0.1 and 0.5.

n	Random		Augmented, $\varepsilon = .5$		Augmented, $\varepsilon = .1$	
	Average	Max	Average	Max	Average	Max
5	1.301	2.875	3.645	4.000	3.627	4.000
7	1.480	2.479	4.545	5.738	4.560	5.879
20	1.854	2.618	4.281	5.090	4.131	5.122
50	1.812	1.971	3.732	3.890	3.633	3.759
100	1.683	1.883	3.567	3.722	3.490	3.812

Compared to the standard random generated instances, the average costs were
almost tripled while the maximum costs were almost doubled. The numerical results
obtained are very interesting since they show that standard random instances are
not really representative when studying the bounds of the MST heuristic for the
MEBR problem. Moreover, as a "side effect" of such experiments, another very
interesting property concerns the topologies obtained in the augmented instances.
Whereas for instances up to around 20 nodes the method modifies the distribution of
nodes, collapsing them to the hexagon shape of Figure 13.1, increasing the number
of nodes makes things more interesting.

In Figure 13.5 an instance of 100 nodes is given before and after the movements
and deletions. What follows from those experiments is an evident regularity on the
final obtained instances. As shown in Figure 13.5, in general, after the augmentation,
nodes look like they are being disposed of on some kind of regular grid, and this
reflects the lower bound given by the regular hexagon shape.

13.2.5 Solving More General Instances of MEBR

In non-geometric versions of MEBR, the MST heuristic can be easily shown to have
a poor approximation ratio. Better algorithms exist in both cases, where the weight

(a) before   (b) after
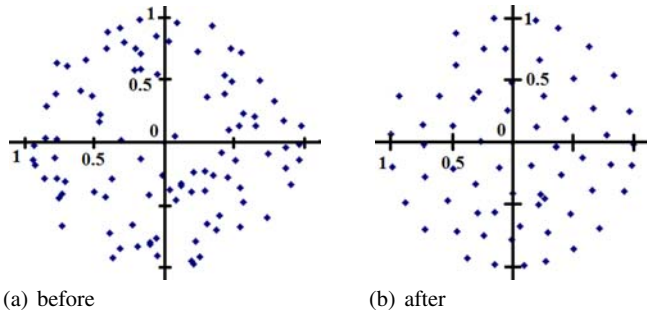
Fig. 13.5 A random instance of 100 nodes before and after applying the augmenting method. The number of nodes decreased from 100 to 65, while the cost increased from 1.877 to 3.681

function is symmetric and nonsymmetric. For the former case, [13–15] present algorithms with a logarithmic (in the number of nodes) approximation ratio. The main idea is to reduce instances of the problem to instances of the problem Node-Weighted Connected Dominating Set [13, 14] or Node-Weighted Steiner Tree [15]. Such instances can be approximated within a logarithmic factor [28], which carries over as the approximation factor of the original instance. For nonsymmetric instances, [15] exploits a reduction due to Liang [36] of instances of MEBR to instances of Directed Steiner Tree. The obtained instances have some special properties, which allow for logarithmic approximations using techniques of Zosin and Khuller [45] for approximating special instances of Directed Steiner Tree. Similar results using different techniques have been obtained independently by Calinescu et al. [9]. All these results are the best possible; a matching inapproximability result has been presented in [19].

In the following, we discuss a recent algorithm from [12] for symmetric instances of MEBR, which has important implications for the geometric version of MEBR as well. The algorithm starts with the solution computed by the MST heuristic and gradually performs improvements on this solution according to a well-selected criterion. At the end, the solution obtained has significantly smaller cost than the initial one.

Before presenting the algorithm, we give some necessary definitions. Given a power assignment p and a station $x \in S$, let $E(p,x) = \{(x,y)|w(x,y) \leq p(x)\}$ be the set of the undirected edges induced by p at x, and $E(p) = \bigcup_{x \in S} E(p,x)$ the set of all the undirected edges induced by p. For every subset of undirected edges $F \subseteq E$ of a weighted graph $G = (S,E)$, we denote as $c(F)$ the overall cost of the edges in F, that is, the total sum of their weights. For the sake of simplicity, we will identify trees with their corresponding sets of edges. A *swap set* for a spanning tree T of an undirected graph $G(S,E)$ and a set of edges F with endpoints in S is any subset F' of edges that must be removed from the multigraph $T \cup F$ so that $T \cup F \setminus F'$ is a spanning tree of G.

We are now ready to describe the algorithm by first giving the basic underlying idea. Starting from a spanning tree T of $G(S)$, if the cost of T is significantly higher

than the one of an optimal solution for performing broadcasting from a given source node $s \in S$, then there must exist a cost-efficient *contraction* of T. Namely, it must be possible to set the transmission power $p(x)$ of at least one station x in such a way that $p(x)$ is much lower than the cost of some swap set $A(p,x)$ for T and $E(p,x)$. The algorithm then repeatedly chooses at each step $p(x)$ in such a way that, starting from the current spanning tree, $c(A(p,x))/p(x)$ is maximized. The final tree will be such that, considering the correct orientation of the edges according to the final assignment p, some edges will be in the reverse direction, i.e., from the leaves towards the source s. However, the transmission powers can then be properly set with low additional cost in order to obtain the right orientation from s to the other stations.

At any intermediate step of the algorithm in which p and T are the current power assignment and maintained tree, respectively, consider a contraction at a given station x consisting of setting the transmission power of x to $p'(x)$, and let p' be the resulting power assignment. Then, a maximum cost swap set $A(p',x)$ to be attributed to the contraction can be trivially determined by letting $A(p',x)$ contain the edges that are removed when determining a minimum spanning tree in the multigraph $T \cup E(p',x)$ with the cost of all the edges in $E(p',x)$ set equal to 0. We call the ratio $\frac{c(A(p',x))}{p'(x)}$ the *cost-efficiency* of the contraction.

Formally, the algorithm performs the following steps:

- Set the transmission power $p(x)$ of every station in $x \in S$ equal to 0.
- Let $T = T(S)$ be a minimum spanning tree of $G(S)$.
- While there exists at least one contraction of cost-efficiency strictly greater than 2

  - Perform a contraction of maximum cost-efficiency, and let $p'(x)$ be the corresponding increased power at a given station x, and p' be the resulting power assignment.
  - Set the weight of all the edges in $E(p',x)$ equal to 0.
  - Let $T' = T \cup E(p',x) \setminus A(p',x)$.
  - Set $T = T'$ and $p = p'$.

- Orient all the edges of T from the source s toward all the other stations.
- Return the transmission power assignment p that induces such a set of oriented edges.

For any instance of the problem where the minimum spanning tree of the cost graph $G(S)$ is guaranteed to cost at most ρ times the cost of an optimal solution for MEBR, the algorithm achieves an approximation ratio bounded by ρ if $\rho \leq 2$ and by $2\ln\rho - 2\ln 2 + 2$ if $\rho > 2$, which exponentially improves upon the MST heuristic. Surprisingly, the algorithm and analysis do not make use of any geometric arguments, and still the results significantly improve the previously best-known approximation factor for Euclidean instances of the problem. The corresponding approximation ratio is reduced (when $\alpha \geq d$) from 6 [1] to 4.2 for $d = 2$, from 18.8 [38] to 6.49 for $d = 3$, and in general from $3^d - 1$ [24] to $2.2d + 0.61$ for $d > 3$. In the two-dimensional case, the achieved approximation is even less than the lower bound of $13/3$ on the approximation ratio of the BIP heuristic [41]. In arbitrary

(i.e., non-Euclidean) cost graphs, it is not difficult to see that the cost of the minimum spanning tree is at most $n - 1$ times the cost of an optimal solution for MEBR; hence, the algorithm achieves a logarithmic approximation for arbitrary symmetric weight functions matching the results in [9, 15].

13.3 Cost Minimization in Multi-interface Networks

Nowadays wireless devices hold multiple radio interfaces, allowing switching from one communication network to another according to required connectivity and related quality. The selection of the "best" radio interface for a specific connection might depend on various factors, namely, its availability in specific devices, the required communication bandwidth, the cost (in terms of energy consumption) of maintaining an active interface, the available neighbors, and so forth. While managing such connections, a lot of effort must be devoted to energy consumption issues. Devices are, in fact, usually battery-powered and network survivability might depend on their persistence in the network. This introduces a challenging and natural optimization problem that must take care of different variables at the same time. Generally speaking, given a set of k interfaces and a graph $G = (V, E)$, where V represents the set of wireless devices and E the set of required connections according to the proximity of the devices and the available interfaces that they may share, the problem can be stated as follows. What is the cheapest way, i.e., which subset of available interfaces in each node must be activated, to satisfy (cover) all the connections described by E? Note that a connection is satisfied when the endpoints of the corresponding edge share at least one active interface. Moreover, for each node $v \in V$ there is a set of available interfaces, from now on denoted as $W(v)$. $\bigcup_{v \in V} W(v)$ determines the set of all the possible interfaces available in the network whose cardinality is denoted by k. An example of a network instance is shown in Figure 13.6.

Depending on whether k is a priori bounded or not, two different problems arise. The first one is called Cost Minimization in Multi-interface Networks (k-CMI for short). The second one is called Cost Minimization in Unbounded Multi-interface Networks (CMI for short). In this section, we report results about the complexity of both k-CMI and CMI in various scenarios. The problems turn out to be very hard in general; hence, we also consider possible approximation algorithms. We deal with two main variations of the problem: the case in which the cost of activating an interface is the same for each interface (uniform case), and the more general case in which such a cost may differ (non-uniform case). Indeed, the first model is equivalent to asking for the minimum total number of activated interfaces inside the network to cover all the connections. We also consider different graph classes that are of interest from both theoretical and practical points of view, namely, graphs with bounded degree, since in real-world scenarios users are normally connected to a limited number of nodes; planar graphs, since the induced graph of joining users in a network is likely to be planar; trees, since middleware strategies are heavily based on this kind of structure (see, for instance, [10]); and complete graphs, since
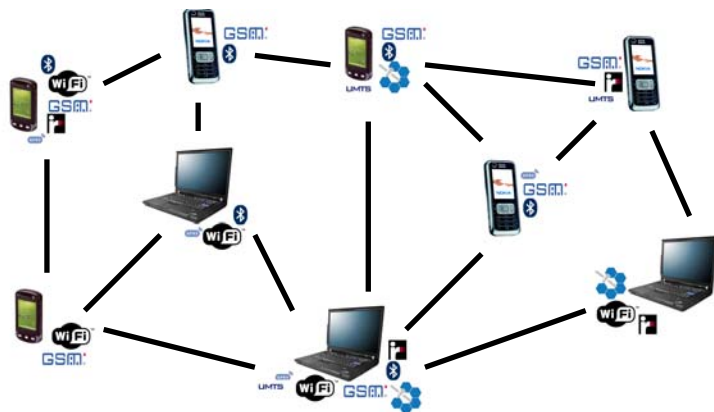
Fig. 13.6 The composed network according to available interfaces and proximities

this is one of the main structures used for modeling peer-to-peer networks (see for instance [18]).

Here we consider the bounded and the unbounded version of the problem. The two models reflect two different feasible cases, where available interfaces are either known a priori or not, respectively. Since nowadays devices support many and different interfaces, it makes sense either to assume the number of interfaces that may occur in a composed network as given, or to not. It might depend, in fact, on the number of nodes participating in the network. Regardless, k reflects the network dynamics.

The problems originated from [11], where a slightly different model of k-CMI is introduced. That model considers bandwidth constraints and also the possibility of having mutually exclusive interfaces, i.e., interfaces that, if activated, preclude the activation of some other interfaces. The motivation is quite technical. For instance, the WiFi interface can operate in different modalities: *Infrastructure* and *Ad Hoc*. If a device activates WiFi in the Infrastructure modality, it cannot satisfy connections that require Ad Hoc modality, and vice versa. This further constraint is not introduced here since the problem, although of practical interest, is not easily solvable. Other related problems were recently addressed in [23, 35] and [4], concerning connectivity and shortest path issues, respectively.

13.3.1 Definitions and Notation

Unless otherwise stated, the network graph $G = (V, E)$ is always assumed to be simple (i.e., without multiple edges), undirected, and connected. Moreover, we always denote by n and m the cardinality of the sets V and E respectively. The degree of

node $v \in V$ is denoted by $deg(v)$ and the set of its neighbors by $N(v)$. The maximum node degree of graph G is denoted by $\Delta(G)$.

A global characterization of interfaces of respective nodes from V is given in terms of an appropriate interface assignment function W, according to the following definition.

Definition 13.1. A function $W: V \to 2^{\{1,\dots,k\}}$ is said to *cover* graph $G = (V, E)$ if for each $\{u, v\} \in E$ the set $W(u) \cap W(v) \neq \emptyset$.

The cost of activating an interface for a node is assumed to be identical for all nodes and given by cost function $c: \{1, \dots, k\} \to \mathbb{Z}_+$, i.e., the cost of interface i is written as c_i. The considered k-CMI optimization problem is formulated as follows.

---

k-CMI: Cost Minimization in Multi-interface Networks

Input: A graph $G = (V, E)$, a positive integer k, an allocation of available interfaces $W: V \to 2^{\{1,\dots,k\}}$ covering graph G, an interface cost function $c: \{1, \dots, k\} \to \mathbb{R}_+$.

Solution: An allocation of active interfaces $W_A: V \to 2^{\{1,\dots,k\}}$ covering graph G such that $W_A(v) \subseteq W(v)$ for all $v \in V$.

Goal: Minimize the total cost of the active interfaces, $c(W_A) = \sum_{v \in V} \sum_{i \in W_A(v)} c_i$.

The considered CMI optimization problem is formulated as follows.

CMI: Cost Minimization in Unbounded Multi-interface Networks

Input: A graph $G = (V, E)$, an allocation of available interfaces $W: V \to 2^{\{1,\dots,k\}}$ covering graph G, an interface cost function $c: \{1, \dots, k\} \to \mathbb{R}_+$.

Solution: An allocation of active interfaces $W_A: V \to 2^{\{1,\dots,k\}}$ covering graph G such that $W_A(v) \subseteq W(v)$ for all $v \in V$.

Goal: Minimize the total cost of the active interfaces, $c(W_A) = \sum_{v \in V} \sum_{i \in W_A(v)} c_i$.

13.3.2 Results for k-CMI

Table 13.2 summarizes known results for k-CMI [32]. The problem is polynomially solvable for $k = 2$ but it is already APX-hard when k grows. If the underlying graph is complete or a tree, then k-CMI is still polynomial while for planar graphs it is NP-hard but admits a PTAS.

Table 13.2 Hardness and approximability of the k-CMI problem

Graph class	Interfaces	Complexity of k-CMI	
		non-uniform costs	uniform costs
General graphs	$k = 2$	$\mathscr{O}(n^3)$	$\mathscr{O}(nm)$
	$k \geq 3$	$(k-1)$-approx,	$\min\{\lceil\frac{k+1}{2}\rceil, \frac{2m}{n}\}$-approx,
		APX-hard	APX-hard
Graphs of bounded	$k \geq 3$	Δ-approx,	$\frac{\Delta+1}{2}$-approx,
degree Δ		APX-hard for $\Delta \geq 5$	APX-hard for $\Delta \geq 5$
Planar graphs	$k \geq 3$	NP-hard, PTAS	NP-hard, PTAS
Trees	any k	$\mathscr{O}(n)$	$\mathscr{O}(n)$
Complete graphs	any k	$\mathscr{O}(n^2)$	$\mathscr{O}(n^2)$

The proof that provides the APX-hardness for $k \geq 3$ considers a polynomial transformation from the well-known VERTEX COVER problem on subcubic graphs[1] to k-CMI. On those instances VERTEX COVER is known to be APX-hard [39]. The transformation works as follows. Given a subcubic graph $G = (V, E)$, it is known that, in general, its chromatic number is at most 3 [6]. Nodes can then be partitioned into three subsets V_1, V_2, and V_3 according to an optimal coloring in such a way that $V_1 \bigcup V_2 \bigcup V_3 \equiv V$ and for each edge $e = \{x, y\} \in E$, x and y do not belong to the same subset V_i for every $i = 1, 2$, or 3.
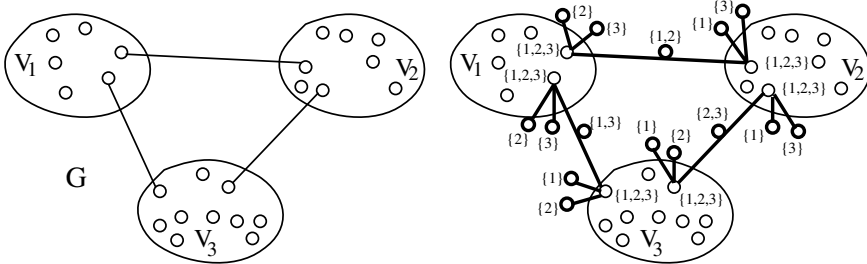
Fig. 13.7 On the left, the graph G subdivided into three node subsets according to a 3-coloring and the three possible kinds of edges. On the right are the modifications obtained for each kind of edge belonging to G and the interfaces associated with the related nodes

As illustrated in Figure 13.7, with each node $v \in V$, three interfaces, namely 1, 2, and 3 are associated. Moreover, to each $v \in V$ there are two new nodes connected. Those new nodes have only one interface: 2 and 3 (1 and 3 or 1 and 2 respectively) if $v \in V_1$ ($v \in V_2$ or $v \in V_3$). For each edge of G a further node is added. With such a node there are associated two interfaces. If the considered edge connects V_1 and V_2 (V_1 and V_3 or V_2 and V_3) then interfaces 1 and 2 (1 and 3 or 2 and 3 respectively) are associated with the added node. Considering for instance an edge $e = \{x, y\} \in E$ such that $x \in V_1$ and $y \in V_2$, in order to solve k-CMI on the new graph of maximum degree 5 built from G, a solution necessarily has to activate interfaces 2 and 3 in

---

[1] Graphs with maximum degree bounded by 3.

x, and 1 and 3 in y. In order for both x and y to be able to communicate with the new intermediate node, either such a node must activate both its interfaces, or one among x and y has to activate its third available interface. Both the solutions are locally equivalent. On the other hand, activating the third interface for either x or y may lead to a decrease in the number of activated interfaces in the global solution. This is implied by the fact that the neighborhood of the added intermediate node between x and y is constituted by only x and y, while both x and y may have many other connections. This implies that one can look for solutions where for each edge of the original graph at least one endpoint has all its three interfaces activated. Note that this reflects exactly the requirement of VERTEX COVER.

k-CMI can be approximated within a factor of $k-1$. A greedy algorithm activates interfaces among the nodes. It starts from the cheapest interface 1, and it activates it in each node that has a neighbor holding that interface. Let $V_1 \subseteq V$ be the set of nodes in which the algorithm activated interface 1 and let $E(V_1)$ be the corresponding set of covered edges. Note that the optimal solution restricted to $E(V_1)$ (i.e., the set of activated interfaces of an optimal solution at the endpoints of the edges belonging to $E(V_1)$) clearly costs at least as much as the cost of the algorithm. In the second step, the same is done for the next cheapest interface 2 among the remaining connections $E \setminus E(V_1)$. Again, the cost of the optimal solution restricted to $E(V_2)$ is at least the price paid by the algorithm. This is implied by the fact that any connection belonging to $E(V_2)$ cannot be covered by interface 1; otherwise, the algorithm would have covered it in the previous step. This process is continued for all the interfaces in a non-decreasing cost order, but for the last two interfaces. Referring to Table 13.2, when $k=2$, k-CMI is polynomially solvable. Hence, when the two most expensive interfaces remain, the optimal algorithm for $k=2$ can be applied. Since each step costs at most as much as the optimal solution, the $(k-1)$-approximation holds by observing that the whole process requires $k-1$ steps.

Concerning the uniform cost case, an easy approximation algorithm for solving k-CMI leads to a factor of $\frac{2m}{n}$. The algorithm simply chooses one interface for each edge of the input graph in order to satisfy the required connection. This means that for each edge at most one interface in each endpoint is activated. It follows that for m edges it activates at most $2m$ interfaces for n nodes. The $\lceil \frac{k+1}{2} \rceil$-approximation mentioned in Table 13.2 is instead obtained by suitably applying a hitting set algorithm.

13.3.3 Results for CMI

Table 13.3 summarizes results obtained for CMI [34]. When k depends on the instance, i.e., it is not set a priori, the problem becomes harder even for complete graphs and trees. In general, CMI is hard to approximate within a factor of $\mathcal{O}(\log k)$, even when restricted to the unit cost interface case. The proof proceeds by reduction to the MINIMUM HITTING SET problem. We recall that for a collection of non-empty subsets $C_1, C_2, \ldots, C_l \subseteq \{1, 2, \ldots, k\}$, set $S \subseteq \{1, 2, \ldots, k\}$ is called a *hitting*

set if for all $i \in [1, \ldots, \ell]$, $C_i \cap S \neq \emptyset$. The problem of minimizing the cardinality of the hitting set is as hard as the MINIMUM SET COVER problem [3], and consequently, hard to approximate within a factor of $\mathscr{O}(\log k)$ [40].

Concerning the $\sqrt{n}(1 + \ln n)$-approximation factor, this is obtained by means of a polynomial transformation of the problem to the well-known WEIGHTED MINIMUM SET COVER problem. This leads to the claim that the existence of any a-approximation algorithm for WEIGHTED MINIMUM SET COVER leads to an $(a\sqrt{n})$-approximation algorithm for CMI. Since WEIGHTED MINIMUM SET COVER admits a $(1 + \ln n)$-approximation [17], $\sqrt{n}(1 + \ln n)$ is obtained.

Table 13.3 Hardness and approximability of the CMI problem. Entries marked by (*) follow from k-CMI results

Graph class	Complexity of CMI	
	non-uniform costs	uniform costs
General graphs	$(k-1)$-approx (*)	$\lceil \frac{k+1}{2} \rceil$-approx (*)
	$(\sqrt{n}(1+\ln n))$-approx	$\frac{2m}{n}$-approx (*)
	not approx within $\mathscr{O}(\log k)$	not approx within $\mathscr{O}(\log k)$
Graphs of bounded degree Δ	Δ-approx (*)	$\frac{\Delta+1}{2}$-approx (*)
	APX-hard for $\Delta \geq 5, k \geq 3$ (*)	APX-hard for $\Delta \geq 5, k \geq 3$ (*)
Planar graphs	6-approx	6-approx
	APX-hard	APX-hard
Trees	2-approx	2-approx
	APX-hard	APX-hard
Complete graphs	not approx within $\mathscr{O}(\log k)$	not approx within $\mathscr{O}(\log k)$

13.4 Conclusion and Future Work

The chapter surveys recent results obtained for two interesting problems arising in the field of wireless ad hoc networks. Both the problems deal with the minimization of the overall energy needed to perform desired communication protocols. In particular, the Minimum Energy Broadcast Routing problem expresses the necessity to perform the basic broadcast pattern of communication from a given source, and the network is composed of homogeneous nodes equipped with omnidirectional radio antennas. The Cost Minimization in Multi-interface Networks expresses the need of establishing connections among heterogeneous nodes equipped with different subsets of interfaces, each associated with some activation cost. Many interesting directions for future work arise from both problems. These include the extensions of the studies to different communication protocols, to different objective functions, and to distributed environments.

References

1. Ambühl, C.: An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP), LNCS 3580, Springer, pp. 1139–1150, 2005

2. Athanassopoulos, S., Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Experimental comparison of algorithms for energy-efficient multicasting in ad hoc networks. In: Proceedings of the 3rd International Conference on Ad Hoc Networks & Wireless (ADHOC NOW), LNCS 3158, Springer, pp. 183-196, 2004

3. Ausiello, G., D'Atri, A., Protasi, M.: Structure preserving reductions among convex optimization problems. Journal of Computer and System Sciences, 21(1):136–153, 1980

4. Barsi, F., Navarra, A., Pinotti, M. C.: Cheapest Path in Multi-Interface Networks. In: Proceedings of the 10th International Conference on Distributed Computing and Networking (ICDCN), LNCS, Springer, 2009

5. Biló, V., Flammini, M., Melideo, G., Moscardelli, L., Navarra, A.: Sharing the cost of multicast transmissions in euclidean and general wireless networks. Theoretical Computer Science, 369(1-3):269–284, 2006

6. Brooks, R. L.: On coloring the nodes of a network. In: Proceedings of Cambridge Philosophical Society, 37:194–197, 1941

7. Čagalj, M., Hubaux, J., Enz, C.: Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues. In: Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom), ACM Press, pp. 172–182, 2002

8. Cai, H., Zhao, Y.: On approximation ratios of minimum-energy multicast routing in wireless networks. Journal of Combinatorial Optimization, 9(3):243–262, 2005

9. Calinescu, G., Kapoor, S., Olshevsky, A., Zelikovsky, A.: Network lifetime and power assignment in adhoc wireless networks. In: Proceedings of the 11th Annual European Symposium on Algorithms (ESA), LNCS 2832, Springer, pp. 114–126, 2003

10. Caporuscio, M., Carzaniga, A., Wolf, A. L.: Design and evaluation of a support service for mobile, wireless publish/subscribe applications. IEEE Transactions on Software Engineering, 29(12):1059–1071, 2003

11. Caporuscio, M., Charlet, D., Issarny, V., Navarra, A.: Energetic Performance of Service-oriented Multi-radio Networks: Issues and Perspectives. In: Proceedings of the 6th International Workshop on Software and Performance (WOSP), pp. 42–45. ACM Press, 2007

12. Caragiannis, I., Flammini, M., Moscardelli, L.: An exponential improvement on the mst heuristic for minimum energy broadcasting in ad hoc wireless networks. In: Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP), LNCS 4596, Springer, pp. 447–458, 2007

13. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: New results for energy-efficient broadcasting in wireless networks. In: Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC), LNCS 2518, Springer, pp. 332–343, 2002

14. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: A logarithmic approximation algorithm for the minimum energy consumption broadcast subgraph problem. Information Processing Letters, 86(3):149-154, 2003

15. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Energy-efficient wireless network design. Theory of Computing Systems, 39(5), pp. 593–617, 2006

16. Cartigny, J., Simplot, D., Stojmenovic, I.: Localized minimum-energy broadcasting in ad hoc networks. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Vol. 3, IEEE Computer Society Press, pp. 2210–2217, 2003

17. Chvátal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research, 4(3):233–235, 1979

18. Cilibrasi, R., Lotker, Z., Navarra, A., Perennes, S., Vitanyi, P.: About the lifespan of peer to peer networks. In: Proceedings of the 10th International Conference On Principles Of Distributed Systems (OPODIS), LNCS 4305, Springer, pp. 290–304, 2006

19. Clementi, A. E. F., Crescenzi, P., Penna, P., Rossi, G., Vocca, P.: On the complexity of computing minimum energy consumption broadcast subgraph. In: Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 2010, Springer, pp. 121–131, 2001

20. Clementi, A. E. F., Di Ianni, M., Silvestri, R.: The minimum broadcast range assignment problem on linear multi-hop wireless networks. Theoretical Computer Science, 299(1-3):751–761, 2003

21. Conway, J. H., Sloane, N. J. A.: "The Kissing Number Problem" and "Bounds on Kissing Numbers". Ch. 2.1 and Ch. 13 in: Sphere Packings, Lattices, and Groups. Springer-Verlag, New York, 3rd edition, 1998

22. Das, A. K., Markas, R. J., El-Sharkawai, M., Arabshahi, P., Gray, A.: Minimum energy broadcast trees for wireless networks: Integer programming formulations. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFO-COM), IEEE Computer Society, Vol. 2, pp. 1001–1010. 2003

23. Faragó, A., Basagni, S.: The Effect of Multi-Radio Nodes on Network Connectivity—A Graph Theoretic Analysis. In: Proceedings of the 19th International IEEE Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2008

24. Flammini, M., Klasing, R., Navarra, A., Perennes, S.: Improved approximation results for the Minimum Energy Broadcasting Problem. In: Proceedings of ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), pp. 85–91, 2004

25. Flammini, M., Klasing, R., Navarra, A., Perennes, S.: Tightening the upper bound for the Minimum Energy Broadcasting. Wireless Networks, Vol. 14(5), pp. 959-669, 2008

26. Flammini, M., Navarra, A., Perennes, S.: The "real" approximation factor of the MST heuristic for the minimum energy broadcasting. ACM Journal of Experimental Algorithmics, 11, 2006. Preliminary version in: Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA), LNCS 3503, Springer, pp. 22–31, 2005

27. Frieze, A. M., McDiarmid, C. J. H.: On Random Minimum Length Spanning Trees. Combinatorica, 9:363–374, 1989

28. Guha, S., Khuller, S.: Improved Methods for Approximating Node Weighted Steiner Trees and Connected Dominating Sets. Information and Computation, 150(1), pp. 57–74, 1999

29. Hac, A.: Wireless sensor network designs. John Wiley & Sons, Ltd, 2003

30. Kang, I., Poovendran, R.: Iterated local optimization for minimum energy broadcast. In: Proceedings of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), pp. 332–341, 2005

31. Klasing, R., Flammini, M., Navarra, A., Perennes, S.: Improved approximation results for the Minimum Energy Broadcasting Problem. Algorithmica, 49(4):318–336, 2007

32. Klasing, R., Kosowski, A., Navarra, A.: Cost minimisation in multi-interface networks. In: Proceedings of the 1st EuroFGI International Conference on Network Control and Optimization (NET-COOP), LNCS 4465, Springer, pp. 276–285, 2007

33. Klasing, R., Navarra, A., Papadopoulos, A., Perennes, S.: Adaptive Broadcast Consumption (ABC), a new heuristic and new bounds for the minimum energy broadcast routing problem. In: Proceedings of the 3rd IFIP-TC6 International Networking Conference, LNCS 3042, Springer, pp. 866–877, 2004

34. Kosowski, A., Navarra, A.: Cost minimisation in unbounded multi-interface networks. In: Proceedings of the 2nd PPAM Workshop on Scheduling for Parallel Computing (SPC), Lecture Notes in Computer Science 4967, Springer-Verlag, pp. 1039-1047, 2007

35. Kosowski, A., Navarra, A., Pinotti, M. C.: Connectivity in Multi-Interface Networks. In: Proceedings of the 4th International Symposium on Trustworthy Global Computing (TGC), LNCS, Springer, 2008
36. Liang, W.: Constructing minimum-energy broadcast trees in wireless ad hoc networks. In: Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), ACM Press, pp. 112–122, 2002
37. Li, F., Nikolaidis, I.: On minimum-energy broadcasting in all-wireless networks. In: Proceedings of the 26th Annual IEEE Conference on Local Computer Networks (LCN), IEEE Computer Society, p. 193, 2001
38. Navarra, A.: 3-D Minimum Energy Broadcasting problem. Ad Hoc Networks, Vol. 6(5), pp. 734-743, 2008
39. Papadimitriou, C. H., Yannakakis, M.: Optimization, approximation, and complexity classes. Journal of Computer and System Sciences, 43:425–440, 1991
40. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In: Proceedings of the 29th Anuual ACM Symposium on Theory of Computing (STOC), pp. 475–484, 1997
41. Wan, P. J., Calinescu, G., Li, X., Frieder, O.: Minimum energy broadcasting in static ad hoc wireless networks. Wireless Networks, 8(6):607–617, 2002
42. Wieselthier, J. E., Nguyen, G. D., Ephremides, A.: On the construction of energy-efficient broadcast and multicast trees in wireless networks. In: Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), IEEE Computer Society, pp. 585–594, 2000
43. Yuan, D.: Computing Optimal or Near-Optimal Trees for Minimum-Energy Broadcasting in Wireless Networks. In: Proceedings of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), pp. 323–331, 2005
44. Zhao, F., Guibas, L.: Wireless sensor networks: an information processing approach. Morgan Kaufmann, 2004
45. Zosin, L., Khuller, S.: On Directed Steiner Trees. In: Proceedings of the 13th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA), pp. 59-63, 2002

Chapter 14
Data Gathering in Wireless Networks

Vincenzo Bonifaci, Ralf Klasing, Peter Korteweg, Leen Stougie, and Alberto Marchetti-Spaccamela

Abstract In this chapter, we address the problem of gathering information in a specific node of a radio network when interference constraints are present. Nodes can communicate data using a radio device; we consider a synchronous time model, where time is divided into rounds. The interference constraints limit the possibility of simultaneous data communication of nodes to the same region of the network. The survey focuses on two interference models, the general interference model and the distance-2 interference model. We survey recent complexity results and approximation algorithms for several variants of the problem. We consider several interference scenarios, the uniform and non-uniform data models, different optimization parameters, and the off-line and online settings of the problem. The objective functions we consider are the minimization of maximum completion time, maximum flow time, and average flow time.

Vincenzo Bonifaci

Dipartimento di Ingegneria Elettrica e dell'Informazione, Università degli Studi dell'Aquila, Poggio di Roio, 67040 L'Aquila, Italy, and

Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma, Italy, e-mail: `bonifaci@dis.uniroma1.it`

Ralf Klasing
CNRS - LaBRI - Université Bordeaux 1, France, e-mail: `klasing@labri.fr`

Peter Korteweg
Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands, e-mail: `peterkorteweg@hotmail.com`

Leen Stougie
Department of Economics and Business Administration, Free University, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands, and
CWI, Amsterdam, The Netherlands, e-mail: `lstougie@feweb.vu.nl`

Alberto Marchetti-Spaccamela
Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma, Italy, e-mail: `alberto@dis.uniroma1.it`

Key words: data gathering, packet radio network, interference, completion time, flow time, approximation algorithm, online algorithm

14.1 Introduction

The wireless gathering problem was proposed by FRANCE TELECOM in the context of providing wireless Internet access to villages [12]. The houses of a village are equipped with a computer, and the computers are interconnected through a wireless local network. To provide Internet access to each of the houses, the computers have to send (and receive) information to a gateway, or *sink node*, which connects the village with the Internet. This creates a special many-to-one information flow demand in which access to the gateway must be provided through multi-hop communication. The radio transmissions which are necessary for data communication are subject to different interference constraints. We are interested in providing interference-free data gathering, minimizing a function of the time required to do so. The underlying problem of gathering data under interference constraints is a fundamental problem in wireless communication, and is also an important building block in more complex communication problems [1, 4]. We call this class of problems *wireless gathering problems* (WGPS). In this chapter we present an overview of recent models and results that are related to WGP.

First, we briefly describe several important features which influence models for wireless gathering in radio networks. These features are all related to the use of radio signals to communicate data, which distinguishes WGP from gathering problems in wired networks.

In radio networks nodes communicate with each other using a radio transmitter. A node broadcasts data to a region surrounding its radio transmitter; the radio signals are transmitted at a certain frequency or within a certain range of frequencies, called the broadcast channel. We restrict our discussion to WGP models with a single broadcast channel.

There are two types of transmitters, based on the antennas being either *unidirectional* or *omnidirectional*. In the omnidirectional case the signal is broadcast in every direction. In this case, under ideal circumstances, the broadcast region can be described as a ball centered at the sender node. In the unidirectional case, the antenna is pointed in a specific direction; hence, the broadcast region can be described as a narrow cone centered at the sender node.

There are two models for radio communication. There is the *half-duplex* model, in which at any instant a node can either send or receive data, and there is the *full-duplex* model, in which a node can both send and receive data simultaneously. We only consider WGP in the half-duplex model. When two nodes communicate, we assume that there is a *sender node*, which has data to send, and a *receiver node* which wishes to receive the data. Data is communicated from the sender node to the receiver node, but the receiver node may use acknowledgement packets (ACKs) to confirm the data reception.

Radio signals have two important properties: fading and interference. *Fading* is the effect of radio signal loss due to physical circumstances. These circumstances are the composition of the space between the sender and receiver nodes, e.g., free space or obstacles, and the distance between the sender and receiver nodes. The strength of a radio signal is a decreasing function of the distance d between the sender node and the receiver node, and the function is in the order of d^{-2} to d^{-6} [1, 18, 37]. For a transmitter to receive data, the radio signal should be of a certain strength. As a consequence of this minimum signal strength and fading, the reachable broadcast region can be described as a closed ball centered at the sender node, where the radius of the ball is called the *communication radius*.

Interference, also called *collision*, is the effect of radio signal loss due to the fact that multiple nodes communicate simultaneously on the same broadcast channel, within the same geographical region. As with data communication, interference occurs if the radio signal is strong enough. When a node broadcasts data, its radio signal is propagated to a region surrounding this node. This interference region is a closed ball centered at the sender node, similarly to the transmission region. We call the radius of this ball the *interference radius*. Note that if a node sends data at a certain power, the interference radius is at least as large as the communication radius, but may be larger, with typical factors between 2 and 3 [22, 37]. A typical assumption is that if a node receives signals from multiple nodes, all data is lost. In some scenarios it could be possible to detect that a collision has occurred, but in this chapter we will assume that no such a collision detection mechanism is available.

The properties of fading and interference highly influence the design of wireless networks and communication algorithms. On the one hand, fading makes communication over long distances costly, and interference limits the data throughput of the network. On the other hand, fading allows multiple nodes to use the same broadcast channel simultaneously, as long as the receiver nodes are sufficiently far apart. This is known as spatial frequency reuse [33, 34].

We assume that not all nodes can communicate directly with the sink, either due to physical constraints or because such communication is too costly in terms of energy usage. We assume nodes use multi-hop communication to communicate data to the sink node. We also assume that the routing network is given. Typically, this routing network is set up via some distributed algorithm [32].

Another feature of many problems is the distinction between a *uniform* and a *non-uniform* data model [7, 13, 22, 25]. In a uniform data model one assumes that each node has the same demand for data communication, and offers the same supply of data communication. In the case of gathering problems this translates into the assumption that each node, except the sink, has an equal number of packets to send; we focus on the case where each node has exactly one packet to communicate to the sink. A non-uniform data model does not impose any restrictions on data demand. Also, most of the models studied in the literature allow the buffering of packets at each node of the network. For a study of gathering protocols in a model where buffering is not allowed see [9].

We present an overview of recent advances in wireless gathering problems. As even recent literature on wireless networks is vast, we have to limit the scope of

the models that we consider. In particular, we will mostly consider the case of omnidirectional antennas, which is where the interference constraints play a key role. When we consider unidirectional antennas we will explicitly say so.

WGP consists of finding an interference-free schedule, in which packets are sent to the sink as fast as possible. We use completion times and flow times as performance measures for the schedule. A completion time model is appropriate for wireless networks which partition data reception and data communication into two phases [22, 25], while a flow time model is appropriate for wireless networks where data reception and communication occur simultaneously.

We focus on theoretical results, which consist of complexity results and worst-case analyses of algorithms using approximation theory and competitive analysis. For complexity theory we refer to Garey and Johnson [23] and Papadimitriou [31]. For a background on approximation theory see the books of Ausiello et al. [2], Hromkovič [26], and Vazirani [39]. For a background on online algorithms and competitive analysis see Borodin and El-Yaniv [17]. We do not consider any empirical studies.

The outline of this chapter is the following. In Section 14.2 we formulate the basic wireless gathering problem mathematically. In Section 14.3 we analyze the complexity of several variants of the problem. In Section 14.4 we present online algorithms for several variants and we analyze their performance using approximation theory and competitive analysis. In Section 14.5 we summarize the models and results presented in this chapter, and outline some interesting open problems.

14.2 The Mathematical Model

The communication model for the wireless gathering problem is a generalization of the classic packet radio network model [3, 4, 7, 8]. Given are a graph $G = (V, E)$ with $|V| = n$, a sink $s \in V$, and a set of packets $J = \{1, 2, \ldots, m\}$. We assume that each edge has unit length. For each pair of nodes $u, v \in V$ we define the *distance* between u and v, denoted by $d(u, v)$, as the length of a shortest path from u to v in G. We have integers d_T and d_I, for the communication radius and interference radius respectively, where naturally we have $d_I \geq d_T$. Each packet $j \in J$ has a *release node* $v_j \in V$ and a *release date* $r_j \in \mathbb{Z}_+$ at which it enters the network. We consider the case where $r_j = 0$ for all j as a special case, which we refer to as WGP without release dates. If there is a single packet j released at each node $v \in V \setminus \{s\}$, the data is said to be *uniform*; otherwise, it is said to be *non-uniform*.

We assume that time is discrete; we call a time unit a *round*. The rounds are numbered $0, 1, \ldots$. During each round a node may be *sending* a packet, be *receiving* a packet, or be *inactive*. If $d(u, v) \leq d_T$ then u can send some packet j to v during a round. If node u sends a packet j to v in some round, then the pair (u, v) is called a *call* of packet j during that round.

We consider two interference models: the general interference model and the distance-2 interference model. In the general interference model two calls (u, v) and

(u', v') *interfere* if $d(u', v) \leq d_I$ or $d(u, v') \leq d_I$; otherwise, the calls are *compatible* [7, 8, 13]. The case $d_T = d_I = 1$ is a special case [3, 4]. In the distance-2 interference model [29] one assumes unit communication radius, and two calls (u, v) and (u', v') are compatible only if $\min_{x \in \{u,v\}, y \in \{u',v'\}} d(x, y) \geq 2$; that is, nodes involved in different calls should be apart at distance at least 2, so at any given round the set of calls forms a matching in the underlying graph. We observe the following relation between the distance-2 interference model and the general interference model: each feasible distance-2 interference schedule is a feasible general interference schedule for $d_T = 1$ and $d_I = 1$, and each feasible general interference schedule for $d_T = 1$ and $d_I = 2$ is a feasible distance-2 interference schedule.

A solution to WGP is a schedule of compatible calls such that all packets are sent to the sink. In principle, each radio transmission could broadcast the same packet to multiple nodes, but in the gathering problem, having more than one copy of each packet does not help – it suffices to keep the one that will arrive first at the sink. Thus, we assume that at any time there is a unique copy of each packet. Also, we assume that packets cannot be aggregated at nodes.

Given a schedule, let v_j^t be the unique node holding packet j at the beginning of round t. The *completion time* of a packet j is $C_j = \min\{t : v_j^t = s\}$. A packet j can be sent for the first time in round r_j. The *flow time* of a packet j is $F_j = C_j - r_j$. We consider the minimization of $\max_j C_j$, called the *makespan*, the minimization of $\max_j F_j$, and the minimization of $\sum_j F_j$. We refer to WGP minimizing the maximum completion time as CMAX-WGP, to WGP minimizing the maximum flow time as FMAX-WGP, and to WGP minimizing the total or average flow time as FSUM-WGP.

14.3 Complexity and Lower Bounds

We give an overview of complexity results and lower bounds on the competitive ratio for WGP.

14.3.1 Minimizing Makespan

The first NP-hardness proof for CMAX-WGP has been given by Bermond et al. [7, 8] by means of a reduction from a satisfiability problem. Here we give a proof that gives more insight into the graph-theoretical nature of the gathering problem. It is based on a reduction from the well-known NP-hard problem of determining the chromatic number of a graph [23] (a similar proof, but within a more general interference model, has been given by Coleri [20]). The chromatic number of a graph is the minimum number of colors needed to color all vertices of the graph so that no two adjacent vertices have the same color.

CHROMATIC NUMBER
Instance: a graph G and an integer k.
Question: does G have chromatic number at most k?

Theorem 14.1. CMAX-WGP *is* NP-*hard in the general interference model.*

Proof. Consider an instance of CHROMATIC NUMBER, that is, an integer k and a graph G, with vertex set $V(G) = \{v_1, \ldots, v_n\}$. Let H be the graph consisting of n isolated vertices $\{u_1, \ldots, u_n\}$. We construct a graph G' with vertex set $V(G') = V(H) \cup V(G) \cup \{s\}$ and edge set $E(G') = E(G) \cup \{[u_i, v_i] : i = 1, \ldots, n\} \cup \{[v_i, s] : i = 1, \ldots, n\}$ (see Figure 14.1). There is one packet in each vertex of H, the sink is the vertex s, $d_T = 1$, and $d_I = 2$.

We prove the theorem by showing that if G has chromatic number at most k, then there is a schedule for the CMAX-WGP instance on G' with makespan at most $k+n$, while if G has chromatic number at least $k+1$, then every schedule for the CMAX-WGP instance on G' has makespan at least $k+n+1$. The theorem then follows since CHROMATIC NUMBER is NP-hard.

Suppose G has chromatic number at least $k+1$. We claim that at any round, in any schedule, the vertices in $V(G)$ that are acting as receivers must form an independent set in G. To see this, notice that any useful transmission to vertex v_i must come from vertex u_i. But then, if (u_i, v_i) and (u_j, v_j) are compatible calls, it must be the case that $[v_i, v_j]$ is not an edge of G; otherwise, interference would occur. Thus, at least $k+1$ rounds are needed to transmit all the packets to vertices in $V(G)$. Additionally, when a vertex v_i transmits to s, no other vertex v_j can receive a packet, because v_j is at distance 2 from v_i. So, calls of the type (v_i, s) are not compatible with calls of the type (u_j, v_j), and a total of $k+1+n$ rounds is needed to gather all packets.

Assume now that G has chromatic number at most k. In a single round, we can forward from $V(H)$ to $V(G)$ any set of packets that corresponds to an independent set of G. Thus, in k rounds we can forward each packet to a vertex of $V(G)$. The remaining n rounds can be used to collect all packets at the sink, one by one. □
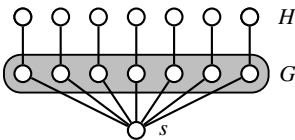
Fig. 14.1 The construction in the proof of Theorem 14.1

The complexity of uniform CMAX-WGP has been analyzed by Korteweg [28]. Kumar et al. [29] presented an inapproximability proof for packet routing in the distance-2 interference model; packet routing is a generalization of WGP in which each packet has to be sent from an arbitrary origin to an arbitrary destination. For both interference models, the NP-hardness of CMAX-WGP can be established by a reduction from the well-known problem of determining the chromatic number of

a graph [23]. The following theorems can be shown to hold using proof ideas and techniques from Korteweg [28] and Kumar et al. [29].

Theorem 14.2. *Uniform* CMAX-WGP *is* NP-*hard.*

Theorem 14.3. CMAX-WGP *is* NP-*hard in the distance-2 interference model.*

We now discuss lower bounds on the approximability and competitive ratio of CMAX-WGP. The following proposition, proved by Korteweg [28], provides a lower bound on the competitive ratio of any online algorithm for CMAX-WGP. Notice that, as is usual in lower bounds for online algorithms, the bound is independent of any hardness assumption.

Proposition 14.1. *No online algorithm for* CMAX-WGP *is better than 7/6-competitive, even if $d_I = d_T$.*

Proof. We give the proof for $d_I = d_T = 1$, the generalization to larger values being straightforward. Consider the graph depicted in Figure 14.2. The adversary releases packet 1 at u at time 0. Observe that for any algorithm that does not send packet 1 in the first round the lemma trivially holds. We assess both deterministic and randomized algorithms by applying Yao's minimax principle [40]. The adversary releases a second packet in round 1 either at u_3 or u_4, each with probability 1/2. Now, the expected number of rounds for any algorithm that sends a packet in the first round is $1/2 \cdot 4 + 1/2 \cdot 3$. In the optimal schedule packet 1 is sent to u_2 (u_1) in the first round if the adversary releases a packet at u_3 (u_4), which yields a makespan of 3. □
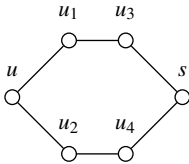
Fig. 14.2 No online algorithm for CMAX-WGP is better than 7/6-competitive ($d_T = d_I = 1$)

It is interesting to note that the example of Proposition 14.1 contains three packets, and there are no known constant lower bound results which hold for instances with an arbitrary number of packets.

Bonifaci et al. provided lower bounds on the approximability of shortest paths following algorithms [13]. A *shortest paths following algorithm* is an algorithm where each packet is sent over some shortest path towards the sink. This is a natural class of algorithms for routing problems, and in case of packet routing without interference it has been demonstrated that for some well-known greedy algorithms there is a gap between the completion times of routing over arbitrary paths and over shortest paths, in favor of routing over shortest paths [19]. The algorithms for WGP that we describe in the next section are shortest paths following. Following [13], for

such algorithms we present a lower bound of $4 - 16/(m+4)$ on their approximation ratio for solving CMAX-WGP on m packets using a shortest paths following algorithm, in the case where $d_T = 1$ and $d_I = 2$.

Consider Figure 14.3. The nodes $u_1, \ldots, u_m$ have one packet each. Any shortest paths following algorithm sends all packets via u, yielding $\max_j C_j = 4m$. There is a solution with no packet passing u that implies $\max_j C_j^* \leq 4 + m$. The example can easily be extended for arbitrary $d_T, d_I = 2d_T$, such that no shortest path following algorithm is better than 4-approximate. In Section 14.4 we discuss a matching upper bound.
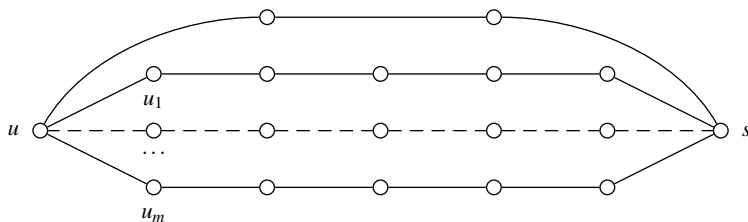
Fig. 14.3 No shortest paths following algorithm is better than 4-approximate for CMAX-WGP ($d_T = 1, d_I = 2$)

14.3.2 Minimizing Flow Times

Bonifaci et al. [15, 16] considered the problems FMAX-WGP and FSUM-WGP. For these versions even stronger results are possible than the one of Theorem 14.1. We present the result for FMAX-WGP.

The lower bound is based on the *induced matching* problem. A matching M in a graph G is an *induced matching* if no two edges in M are joined by an edge of G. The following rather straightforward relation between compatible calls in a bipartite graph and induced matchings will be crucial in the proof.

Proposition 14.2. *Let $G = (U, V, E)$ be a bipartite graph with node sets (U, V) and edge set E. Then, a set $M \subseteq E$ is an induced matching if and only if the calls corresponding to edges of M, directed from U to V, are all pairwise compatible, assuming $d_T = d_I = 1$.*

INDUCED BIPARTITE MATCHING (IBM)
Instance: a bipartite graph G and an integer k.
Question: does G have an induced matching of size at least k?

The proof by Bonifaci et al. uses the fact that the optimization version of IBM is hard to approximate: there exists an $\alpha > 1$ such that it is NP-hard to distinguish between graphs with induced matchings of size k and graphs in which all induced matchings are of size at most k/α. The current best bound for α is 6600/6599 [21].

Theorem 14.4. *Unless* $P = NP$, *no polynomial-time algorithm can have approxima-tion ratio better than* $\Omega(m^{1/3})$ *for* FMAX-WGP *in the general interference model, even when* $d_T = d_I = 1$.

Proof. Let (G,k) be an instance of IBM, $G = (U,V,E)$. We construct a four-layer network with a unique source o (first layer), a clique on U and a clique on V (middle layers), and a sink s (last layer). Source o is adjacent to each node in U, and s to each node in V. The edges between U and V are the same as in G (see Figure 14.4). We set $d_T = d_I = 1$.
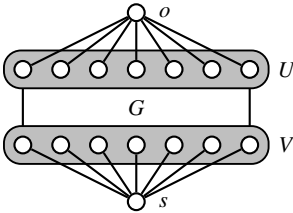
Fig. 14.4 The construction in the proof of Theorem 14.4

The FMAX-WGP instance consists of $m = (1 - 1/\alpha)^{-1}(1 + k/\alpha)(2k + 1)k = \Theta(k^3)$ packets with origin o. They are divided into m/k groups of size k. Each packet in the hth group has release date $(k+1)h$, $h = 0, \ldots, m/k - 1$. Rounds $(k+1)h$ to $(k+1)(h+1) - 1$ together are a *phase*.

We prove that if G has an induced matching of size k, there is a gathering schedule of cost $2k + 1$, while if G has no induced matching of size more than k/α, every schedule has cost at least $(2k+1)k = (2k+1)\Theta(m^{1/3})$. The theorem then follows directly.

Assume G has an induced matching M of size k, say (u_i, v_i), $i = 0 \ldots k - 1$. Then consider the following gathering schedule. In each phase, the k new packets at o are transmitted, necessarily one by one, to layer U while old packets at layer V (if any) are absorbed at the sink; then, in a *single* round, the k new packets move from U to V via the matching edges. More precisely, each phase can be scheduled in $k + 1$ rounds as follows:

1. for $i = 0, \ldots, k - 1$ execute in round i the two calls (o, u_i) and $(v_{i+1 \bmod k}, s)$;
2. in round k, execute simultaneously all the calls (u_i, v_i), $i = 0, \ldots, k - 1$.

The maximum flow time of the schedule is $2k + 1$, as a packet released in phase h reaches the sink before the end of phase $h + 1$.

In the other direction, assume that each induced matching of G is of size at most k/α. By Proposition 14.2, at most k/α calls can be scheduled in any round from layer U to layer V. We ignore potential interference between calls from o to U and calls from V to s; doing so may decrease the cost of a schedule. As a consequence, we can assume that each packet follows a shortest path from o to s. Notice however that, due to the cliques on the layers U and V, no call from U to V is compatible with a call from o to U, or with a call from V to s.

Let m_o and m_U be the number of packets at o and U, respectively, at the beginning of a given phase. Also, let $\beta = 1 + k/\alpha$. We associate with the phase a potential value $\psi = \beta m_o + m_U$, and we show that at the end of the phase the potential will have increased proportionally to k. Let c_o and c_U denote the number of calls from o to U and from U to V, respectively, during the phase. Since a phase consists of $k+1$ rounds, and in each round at most k/α calls are scheduled from U to V, we have $c_o + c_U/(k/\alpha) \leq k+1$, or, equivalently since $k/\alpha = \beta - 1$,

$$(\beta - 1)c_o + c_U \leq (\beta - 1)(k+1). \tag{14.1}$$

If m'_o, m'_U are the number of packets at o and U at the beginning of the next phase, and $\psi' = \beta m'_o + m'_U$ is the new potential, we have

$$
\begin{aligned}
m'_o &= m_o + k - c_o \\
m'_U &= m_U + c_o - c_U \\
\psi' - \psi &= \beta(m'_o - m_o) + m'_U - m_U \\
&= \beta(k - c_o) + c_o - c_U \\
&= \beta k - (\beta - 1)c_o - c_U \\
&\geq \beta k - (\beta - 1)(k+1) \\
&= k - (\beta - 1) \\
&= (1 - 1/\alpha)k
\end{aligned}
$$

where the inequality uses (14.1).

Thus, consider the situation after m/k phases. The potential has become at least $\Psi = (1 - 1/\alpha)m$. By definition of the potential, this implies that at least $\Psi/\beta = (1 - 1/\alpha)(1 + k/\alpha)^{-1}m = (2k+1)k$ packets reside at either o or U; in particular, they have been released but not yet absorbed at the sink. Since the sink cannot receive more than one packet per round, this clearly implies a maximum flow time of $(2k+1)k = (2k+1)\Theta(m^{1/3})$ for one of these packets. $\square$

A similar construction shows that the same problem with minimization of total flow time FSUM-WGP cannot be approximated within a ratio better than $\Omega(m^{1-\varepsilon})$ for any $0 < \varepsilon < 1$ [15]. We also notice that a similar instance as that used in Section 14.3.1 constructed for proving inapproximability of shortest paths following algorithms for CMAX-WGP can be constructed here to prove that shortest paths following algorithms cannot approximate optimal solutions of FMAX-WGP and FSUM-WGP within a ratio better than $\Omega(m)$.

For the distributed model, Bonifaci et al. [14] provided lower bounds for FMAX-WGP which do not depend on the assumption $\mathsf{P} \neq \mathsf{NP}$. They consider a scenario in which the network is partitioned into layers based on distance to the sink. They assume that interference conflicts between transmissions from one layer of the tree to the next are resolved randomly: whenever several transmissions from a layer occur in the same round, only a uniformly chosen one succeeds; this is called the *random selection model*. This assumption seems natural for distributed algorithms, as they have no simple means of coordinating the transmitting nodes (or more precisely,

coordinating the transmitting nodes is as hard as the original communication task). For distributed algorithms following a random selection model they present the following lower bound.

Theorem 14.5. *In the random selection model the approximation ratio of any algorithm for* FMAX-WGP *is at least* $\Omega(\log m)$.

In fact, Bonifaci et al. [14] argue that even resource augmentation using speed as a resource is not likely to improve this lower bound. The reason is that the lower bound is due to an adversarial selection of which packet to advance; and the probability of obtaining such a selection depends on the number of packets, and not on the speed of the algorithm.

14.4 Online Algorithms

14.4.1 Minimizing Makespan

14.4.1.1 Omnidirectional Antennas

Several authors have presented centralized online algorithms for omnidirectional WGP. The first algorithm, PIPELINE, was presented by Bermond et al. [7, 8]. The algorithm was analyzed in an off-line context, but can be implemented in an online setting. The idea of the algorithm is to pipeline packets towards the sink by partitioning the graph into intervals. The lengths of the intervals are chosen such that packets can advance in parallel without interfering with each other.

First, we introduce some notation to facilitate the exposition of the algorithm. An important concept used in this and other algorithms is that of critical radius. The *critical radius* R^* is the greatest integer R such that no two nodes at distance at most R from s can receive a packet in the same round. It is not hard to show that $R^* \geq \lfloor \frac{d_I - d_T}{2} \rfloor$ (see, for example, [7, 8]). The *critical region* is the ball centered at s of radius R^*. Thus, at any round at most one node in the critical region can receive a packet. We define $K^* = \lceil \frac{R^* + 1}{d_T} \rceil \geq 1$ and $K = 1 + \lceil \frac{d_I + 1}{d_T} \rceil$. Roughly stated, K gives an upper bound on the number of rounds during which a packet needs to be forwarded before a new packet can be safely forwarded from the same origin over the same path, while K^* gives a lower bound on the number of rounds during which a packet has to move inside the critical region, assuming it starts outside. We also let $K_0 = 1 + \lceil \frac{R^*}{d_T} \rceil$, $\text{Rad} = \max_{u \in V} d(u, s)$, and $L = 1 + \lceil \frac{\text{Rad} - K_0 d_T}{K d_T} \rceil$.

The algorithm partitions the set of distances to the sink $[1, \text{Rad}]$ into L intervals $I_0, \ldots, I_{L-1}$. These are defined by $I_0 = [1, K_0 d_T]$ and, for $i = 1, \ldots, L-1$, $I_i = [K_0 d_T + 1 + (i-1)K d_T, K_0 d_T + i K d_T]$.

Additionally, each I_i is split into *areas* of length d_T, so I_0 is split into K_0 areas $I_0^j = [K_0 d_T + 1 - j d_T, K_0 d_T - (j-1)d_T]$, $j = 1, \ldots, K_0$; and I_i, $i = 1, \ldots, L-1$ is split into K areas $I_i^j = [K_0 d_T + 1 + i K d_T - j d_T, K_0 d_T + i K d_T - (j-1)d_T]$, $j = 1,$

..., K. We denote the set of vertices whose distance is in I_i (respectively I_i^j) by V_i (respectively V_i^j). Figure 14.5 shows a partition with $K = 4, K_0 = 3, d_T = 2$.
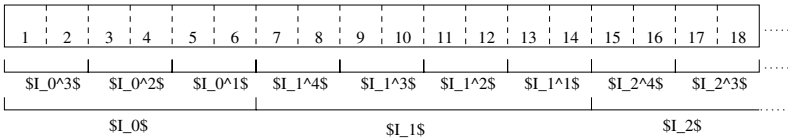
Fig. 14.5 Partitioning of distance intervals for $K = 4, K_0 = 3, d_T = 2$

We are now in position to describe the algorithm (Algorithm 14.1).

Algorithm 14.1 PIPELINE

The algorithm works in phases. Each phase, except possibly the last, consists of K rounds t_j, $j = 1, \ldots, K$. The algorithm uses the concepts of intervals and areas to construct a set of feasible calls in each round.

for each phase **do**
  for each round t_j, $j = 1, 2, \ldots, K$ **do**
    Select in each interval I_i a vertex u_i^j in V_i^j with an available packet to transmit (if such a vertex exists). Vertex u_i^j calls the closest vertex in the preceding area, i.e., if $d(u_i^j, s) = K_0 d_T + 1 + iKd_T - jd_T + \alpha$ for some $0 \le \alpha < d_T$, then u_i^j calls a vertex v such that $d(v, s) = K_0 d_T + iKd_T - jd_T$. This means that if $i = 0$ and $j < K_0$ (or $i > 0$ and $j < K$) then $v \in V_i^{j+1}$, if $i > 0$ and $j = K$ then $v \in V_{i-1}^1$, and if $i = 0$ and $j = K_0$ then $v = s$.

We claim that PIPELINE creates a feasible schedule for WGP. First, let us show that for any round the calls scheduled by PIPELINE are all pairwise compatible. Indeed, consider two calls $(u, v) \ne (u', v')$ of the same round t_j. Then $d(u, s) = K_0 d_T + 1 + iKd_T - jd_T + \alpha$, for some $i \ge 0, 0 \le \alpha < d_T$, and $d(v', s) = K_0 d_T + i'Kd_T - jd_T$ for some $i' \ne i$ (as $v \ne v'$). Therefore, $d(u, v') \ge |(i' - i)Kd_T - 1 - \alpha| \ge d_I + d_T - \alpha \ge d_I + 1$. Similarly one can show $d(u', v) \ge d_I + 1$, and the calls are compatible. To see why the algorithm delivers all the packets, observe that after a phase of K rounds, the protocol ensures that if a vertex of V_i contains a packet, then the last vertex of V_{i-1} has received a new packet.

To illustrate PIPELINE we show one phase of the algorithm in Figure 14.6.

Bonifaci et al. [13] presented a class of online centralized algorithms for CMAX-WGP, called PRIORITY GREEDY. In a PRIORITY GREEDY algorithm each packet is assigned a unique priority based on some algorithm-specific rules, and the priority ordering does not change over time. Then, in each round, packets are considered in order of decreasing priority and are sent towards the sink as far as possible while avoiding interference with higher priority packets. Thus, the schedule output by the algorithm is feasible by construction.
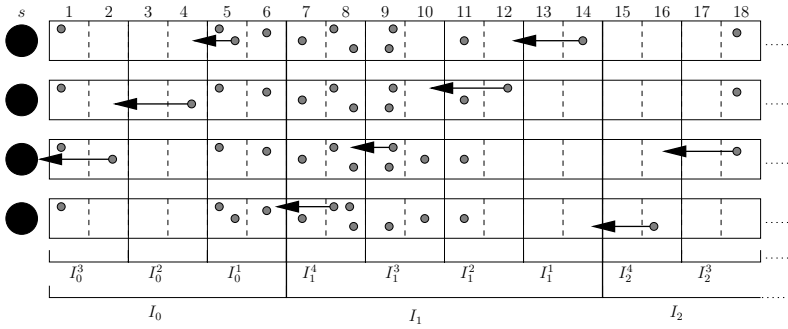
Fig. 14.6 A phase of PIPELINE, consisting of $K = 4$ rounds. Here, packets are represented as small balls. Notice that packets in the same cell are at the same distance from the sink, but they can be in different vertices

---

Algorithm 14.2 PRIORITY GREEDY

---

for each round $t = 0, 1, 2, \ldots$ **do**

Consider the available packets in order of decreasing priority, and send each packet as far as possible along a shortest path from its current node to the sink, without causing interference with any higher-priority packet.

Both Bermond et al. and Bonifaci et al. use similar concepts to derive upper bounds on their algorithms, as well as a lower bound on the makespan of an off-line optimal solution [7, 8, 13].

The lower bound on the completion time of any schedule is based on the observation that at most one packet can be sent from a node within the critical region. Let $\delta_j = \lceil \frac{d(v_j, s)}{d_T} \rceil$, the minimum number of calls required for packet j to reach s. Define also $\pi_j = \min\{\delta_j, K^*\}$ and $R_j = r_j + \delta_j - \pi_j$. Informally, π_j gives the number of rounds during which packet j has to move inside the critical region (irrespective of whether it originated inside or outside of it); R_j is the first possible time at which packet j can reach the border of the critical region. The following bound on the cost of an optimal solution can be proved by considering only the processing that has to be done inside the critical region [13].

Lemma 14.1. *Let $S \subseteq J$ be a nonempty set of packets, and let C_i^* denote the completion time of packet i in some feasible schedule. Then there is $k \in S$ such that* $\max_{i \in S} C_i^* \geq R_k + \sum_{i \in S} \pi_i$.

We sketch the idea behind the upper bound on the completion times; the sketch is based on the upper bound proof of the PRIORITY GREEDY algorithm. The idea is that if a packet is delayed, i.e., it is not sent as far as possible in each round until it reaches the sink, then this packet must be close to some other packet that is sent in that round. As a result we can provide a bound on the completion time of a packet by relating it to the completion time of another packet that delays this packet. This provides an upper bound on the makespan of a set of packets. Formally, we say that packet j is *blocked* in round t if $t \geq r_j$ but j is not sent over distance d_T in round

t. Note that in a PRIORITY GREEDY algorithm a packet can only be blocked due to interference with a higher priority packet. We define the following *blocking relation* on a PRIORITY GREEDY schedule: $k \prec j$ if in the last round in which j is blocked, k is the packet closest to j that is sent in that round and has a priority higher than j (ties broken arbitrarily). The blocking relation induces a directed graph $F = (J, A)$ on the packet set J with an arc (k, j) for each $k, j \in J$ such that $k \prec j$. Observe that, for any PRIORITY GREEDY schedule, F is a directed forest and the root of each tree of F is a packet which is never blocked. For each j, let $T(j) \subseteq F$ be the tree of F containing j, $b(j) \in J$ be the root of $T(j)$, and $P(j)$ be the set of packets along the path in F from $b(j)$ to j.

Lemma 14.2. *For each packet $j \in J$ in a* PRIORITY GREEDY *schedule, $C_j \leq R_{b(j)} + (K/K^*) \cdot \sum_{i \in P(j)} \pi_i$.*

Bonifaci et al. [13] considered a particular PRIORITY GREEDY algorithm called RPG in which packet j has higher priority than packet k if $R_j < R_k$ (ties broken arbitrarily). Combining Lemmas 14.1 and 14.2 they proved the following theorem.

Theorem 14.6. RPG *is K/K^*-competitive for* CMAX-WGP.

Similarly, Bermond et al. [7] presented the following theorem.

Theorem 14.7. PIPELINE *is K/K^*-competitive for* CMAX-WGP *without release dates.*

The exact ratio depends on d_T and d_I, but is always bounded: it is straightforward to verify that $2 \leq K/K^* \leq 4$ for all d_T and d_I, while $K/K^* \leq 3$ for $d_I = d_T$. Similarly, Korteweg [28] proved that PRIORITY GREEDY is $(K/K^* + 1)$-competitive for *any* fixed priority on the packets, using Lemmas 14.1 and 14.2. An interesting open problem is whether there exists a polynomial-time approximation scheme, or a $(1 + \varepsilon)$-approximation algorithm for general graphs for small values of $\varepsilon > 0$.

Notice that the algorithms PIPELINE and PRIORITY GREEDY can be implemented using only local information. Namely, it suffices that a node is informed about the state of nodes within distance $d_I + 1$.

Kumar et al. [29] presented a decentralized algorithm for packet routing under the distance-2 interference model. The authors presented an $\mathcal{O}(\Delta \log^2 n)$-approximation algorithm, where Δ is the maximum graph degree and n is the number of nodes. Their algorithm assumes that each node knows upper bounds on the maximum number of packets per node and the network diameter. The first assumption seems restrictive from a practical point of view, where packets arrive online over time. The algorithm proceeds in phases, and at the start of each phase nodes communicate with nodes up to a distance 3 to determine interference-free schedules for the round in the next phase. As such the algorithm, like those discussed above, is decentralized, but not distributed in the sense that nodes use information about neighboring nodes.

Bar-Yehuda et al. [4] considered a distributed algorithm for CMAX-WGP in the special case where $d_T = d_I = 1$ and there are no release dates. We refer to their algorithm as DISTRIBUTED GREEDY (DG). The idea behind DG is the following.

To reduce interference between nodes, DG partitions nodes into layers, and assigns a label to nodes in a layer. A *layer* is a set of all nodes at the same distance from the sink. A node at distance d from the sink is assigned *label d* mod 3. Each node can be either *active* during a round or *inactive*; only active nodes will transmit a packet. A node will not be active if its packet buffer is empty.

DG uses a procedure to establish communication from a set of active nodes. The procedure, first introduced and studied by Bar-Yehuda, Goldreich, and Itai [3], is called DECAY and requires $2 \log \Delta$ rounds; the time needed for a single execution of the procedure is called a *phase*.

Algorithm 14.3 DECAY(u, v)

---

    for $j = 1, 2, \ldots, 2 \log \Delta$ **do**
        u sends to v the oldest packet from its buffer;
        u deactivates itself for the rest of the phase with probability 1/2.

In fact, the original description does not describe which packet v to advance from the buffer, because for the analysis of completion times the choice of this packet is not relevant. For flow times the choice can be relevant; hence, we choose to advance the oldest packet. We now present the description of the DISTRIBUTED GREEDY algorithm (Algorithm 14.4).

Algorithm 14.4 DISTRIBUTED GREEDY (DG)

---

    for each phase $k = 1, 2, \ldots$ **do**
        Activate each node with label k mod 3 that has a nonempty packet buffer;
        Execute DECAY$(u, \text{parent}(u))$ in parallel for each active node u.

Although the algorithm does not model acknowledgement of packets explicitly, it is easy to include them, e.g., by doubling the number of rounds, having communication in odd rounds and acknowledgements in even rounds, as observed by Bar-Yehuda et al. Using this, we can assume that successful receipt of a packet (by the parent of the sending node in the communication tree) is acknowledged immediately. Only at that time does it get removed from the sender's buffer.

By the transmission protocol in DG, where in phase k only nodes of layer k mod 3 transmit, if two nodes transmit, then either they are at the same layer or they are at least distance 3 apart. Hence, in DG two nodes can only interfere if both sender nodes are in the same layer.

A *superphase* consists of three consecutive phases. Another important ingredient in the analysis of DG is the following, proved by Bar-Yehuda et al. [4].

Theorem 14.8. *Let i be any layer of the tree containing some packet at the beginning of a superphase. There is probability at least $\mu = e^{-1}(1 - e^{-1})$ that during this superphase DG sends successfully a packet from at least one node u in layer i to the parent node of u in the communication tree.*

This theorem shows that, during a superphase, each nonempty layer forwards a packet with probability μ to the following layer. Notice however that there is no guarantee on which particular packet is advanced. The use of superphases and labels, i.e., a synchronous model, is essential to the proof of Theorem 14.8. If the DECAY procedure is applied in an asynchronous model, it is not clear whether a similar constant probability μ is attainable.

Theorem 14.8 suffices to bound the completion time of packets in a schedule constructed by DG.

Theorem 14.9. DISTRIBUTED GREEDY *is in expectation* $\mathcal{O}(\log \Delta)$*-competitive for* CMAX-WGP *without release dates, and* $d_I = d_T = 1$.

14.4.1.2 Special Topologies

The hardness results for CMAX-WGP on general graphs of Section 14.3 motivate the study of specific topologies, such as the path, balanced stars, and the two-dimensional grid. With the additional assumption that the data is uniform (every node holds exactly one packet) it is possible to provide algorithms whose performance differs only by an additive constant from the theoretical minimum. We are not aware of studies for specific topologies in the case of non-uniform data (in particular, it is unknown whether optimal polynomial-time algorithms are possible for path or tree topologies), although it is certainly possible to at least improve the approximation guarantees in this setting.

As an example of the uniform data model on a specific topology, Figure 14.7 shows an optimal gathering schedule using 18 rounds for a path of seven vertices (each having one data packet), with $d_T = 1$, $d_I = 2$, and $s = 0$. The schedule has a regular structure and this regularity can be exploited to give a general algorithm for paths whose cost differs by the optimal one only by an additive constant term (though the constant may depend on d_T and d_I).
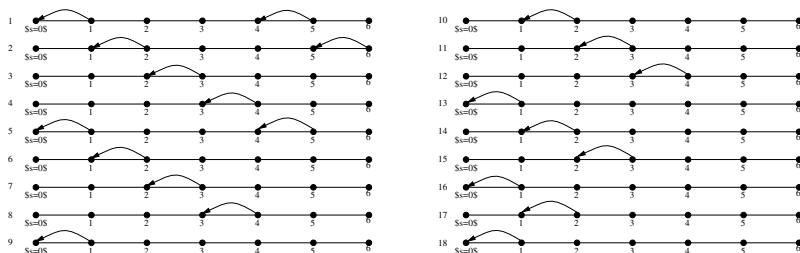
Fig. 14.7 A gathering schedule in the path when $d_T = 1$, $d_I = 2$, and every vertex has one packet to send to the sink $s = 0$

In fact, the uniform model has first been studied in the case of specific graph topologies for specific values of d_I and d_T. In particular, the case $d_T = 1$ was studied in [5] for the case where the graph is a path, and in [10] for the case where the graph

is the two-dimensional square grid. An optimal algorithm for trees when $d_T = d_I = 1$ is given by Bermond and Yu [11].

Bermond et al. [6, 30] consider the uniform model for paths and grids, for any value of d_T. Even though their algorithms do not match the lower bounds, they are again larger by an additive constant that depends only on d_I and d_T. The authors also study the case of stars and show that the general lower bound of [7, 8] is tight up to a constant that does not depend on the size of the network. Table 14.1 shows the main results of Bermond et al. [6, 30]. The notation is the following:

- LB (UB) is a lower bound (upper bound) on the number of rounds for gathering in the corresponding topology.
- P_n is the path with n vertices $0, 1, \ldots, n-1$. Vertex i is adjacent to vertex $i+1$ for any $i = 1, \ldots, n-2$. Therefore, the sink s is simply an integer such that $0 \leq s \leq n-1$.
- $S_{K,l}$ is the balanced spider graph with K branches. $S_{K,l}$ consists of K copies of P_l (called branches) sharing a common extreme, the sink s.
- $G^2(p,q)$ is the two-dimensional grid, i.e., the graph $G = (V,E)$ where $V = \{(i,j) : -p \leq i \leq p, -q \leq j \leq q\}$. So $n = (2p+1)(2q+1)$, and (x,y) and (x',y') are connected when $|x - x'| + |y - y'| = 1$. We assume that $p, q \geq d_I + d_T + 1$ and $s = (0,0)$.

In Table 14.1, $\mathscr{O}(1)$ is used to denote a constant that may depend on d_I and d_T but not on the size n of the network.

Table 14.1 Approximation results for gathering in specific topologies

Topology	LB	UB
P_n	$\frac{d_I+d_T+1}{d_T}\max[s,n-s] - \mathscr{O}(1)$	$\frac{d_I+d_T+1}{d_T}\max[s,n-s] + \mathscr{O}(1)$
$S_{K,l}$, $\lfloor d_I/d_T \rfloor$ odd	$\frac{1}{2}(1+\lfloor d_I/d_T \rfloor)n - \mathscr{O}(1)$	$\frac{1}{2}(1+\lfloor d_I/d_T \rfloor)n + \mathscr{O}(1)$
$S_{K,l}$, $\lfloor d_I/d_T \rfloor$ even	$\frac{1}{2}\lfloor d_I/d_T \rfloor n + \frac{n}{K} - \mathscr{O}(1)$	$\frac{1}{2}\lfloor d_I/d_T \rfloor n + \frac{n}{K} + \mathscr{O}(1)$
$G^2(p,q)$, $\lfloor d_I/d_T \rfloor$ odd	$\frac{1}{2}(1+\lfloor d_I/d_T \rfloor)n - \mathscr{O}(1)$	$\frac{1}{2}(1+\lfloor d_I/d_T \rfloor)n + \mathscr{O}(1)$
$G^2(p,q)$, $\lfloor d_I/d_T \rfloor$ even	$\frac{1}{2}\lfloor d_I/d_T \rfloor n + \frac{n}{4} - \mathscr{O}(1)$	$\frac{1}{2}\lfloor d_I/d_T \rfloor n + \frac{n}{4} + \mathscr{O}(1)$

14.4.1.3 Unidirectional Antennas

We discuss here some results for gathering problems with unidirectional antennas. Florens et al. [22] study makespan minimization in a model where each each node is equipped with directional antennas and has no buffering capacity. Furthermore, it is assumed that a node cannot receive and send simultaneously, that the communication radius is 1, and that there is no interference but each node can only receive one message at a time. Under these assumptions, Florens et al. give optimal (polynomial-time) gathering algorithms for path and tree networks. Their work has

been extended to general graphs in the uniform case by Gargano and Rescigno [25]. Other results for specific topologies are discussed by Revah and Segal [35, 36] and Segal and Yedidsion [38].

A discussion of some algorithmic and graph-theoretic problems related to wireless data gathering with minimum makespan is contained in [24]. Finally, another related model can be found in Klasing et al. [27], where the authors study the case in which steady-state flow demands between each pair of nodes have to be satisfied.

14.4.2 Minimizing Flow Times

Most literature on gathering problems focuses on minimizing completion times. In this subsection we highlight some results on minimizing flow times. First, we consider the centralized model. Bonifaci et al. [16] analyzed FMAX-WGP in the general interference model. For this version they analyzed the performance of a particular PRIORITY GREEDY algorithm. Because it follows from Theorem 14.4 that there is no constant approximation algorithm for this problem, unless $P = NP$, they used resource augmentation to analyze the quality of the algorithm. They study a variant of PRIORITY GREEDY which orders packets based on release dates, i.e., packet j precedes k if $r_j \leq r_k$; ties ($r_j = r_k$) are broken arbitrarily. They call this variant FIFO after the well known first-in-first-out algorithm in scheduling theory, although in this case the term FIFO refers to the priority ordering; observe that the first packet in the system does not have to arrive earliest at the sink using FIFO. They use FIFO as a sub-routine in an algorithm which can be used in a resource augmentation setting based on speed. The algorithm is the so-called σ-speed algorithm, where the parameter σ denotes the ratio between the clock speed of the algorithm and the clock speed of the optimal solution to which the algorithm is compared. The algorithm is the following (Algorithm 14.5).

Algorithm 14.5 σ-FIFO

---

1. Create a new instance $\mathscr{I}'$ by multiplying release dates: $r'_j = \sigma r_j$;
2. Run FIFO on $\mathscr{I}'$;
3. Speed up the schedule thus obtained by a factor of σ.

The schedule constructed by σ-FIFO is a feasible σ-speed solution to the original problem because of step 1. Bonifaci et al. [16] prove that this algorithm is optimal for some σ which depends on K and K^*, but is never larger than 5.

Theorem 14.10. *For* $\sigma \geq K/K^* + 1$, σ-FIFO *is a* σ-speed optimal algorithm for FMAX-WGP *in the general interference model.*

In [15] complementary and indeed similar results have been obtained for the problem with the average completion time as objective, FSUM-WGP.

For the distributed model, for WGP minimizing flow times in the general interference model, the performance of algorithm DG is studied by Bonifaci et al. [14]. Again, the performance of the algorithm was studied in a resource augmentation setting with an increase in speed of factor σ, similarly to the centralized model. We refer to this version as σ-speed DG, although the algorithm is identical to DISTRIBUTED GREEDY. Also, they focused on minimizing average flow times instead of minimizing maximum flow times. The motivation for this objective over the objective minimizing maximum flow times is based on the proof of the lower bound of Theorem 14.5. As described above the proof indicates that for a general class of distributed algorithms, i.e., algorithms which base decisions on random selection, it is rather unlikely that there exists a constant competitive algorithm for this problem, even if one allows resource augmentation using extra speed. The same authors presented the following positive result.

Theorem 14.11. *Let $0 < \varepsilon \le 1$ and $\sigma = 6\mu^{-1} \cdot \log \Delta \cdot \ln(\delta/\varepsilon)$. Then σ-speed DG is in expectation $(1 + 3\varepsilon)$-competitive when minimizing the average flow time.*

14.5 Conclusion

The chapter surveys recent complexity results and approximation algorithms for several variants of the wireless gathering problem. It considers different interference models, the uniform and non-uniform data models, different optimization parameters, and the off-line and online settings of the problem.

Many interesting directions of future work arise from the considered problems. These include the attempt to close the existing gaps between the upper and lower bounds for the specific problems. Where good solutions on general graphs are not possible or not available, the focus on graph classes that are of interest from a practical point of view is of high importance. In the non-uniform data model many important questions are still to be resolved. Also, more work remains to be done on unidirectional antennas with or without buffering capabilities at the nodes. Finally, especially from a practical perspective, the study of distributed algorithms needs to be further intensified.

References

1. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks **38**(4), 393–422 (2002)
2. Ausiello, G., Protasi, M., Marchetti-Spaccamela, A., Gambosi, G., Crescenzi, P., Kann, V.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer (1999)
3. Bar-Yehuda, R., Goldreich, O., Itai, A.: On the time-complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization. Journal of Computer and System Sciences **45**(1), 104–126 (1992)
4. Bar-Yehuda, R., Israeli, A., Itai, A.: Multiple communication in multihop radio networks. SIAM Journal on Computing **22**(4), 875–887 (1993)
5. Bermond, J. C., Corrêa, R. C., Yu, M. L.: Gathering algorithms on paths under interference constraints. In: Proceedings of the 6th Italian Conference Algorithms and Complexity, *Lecture Notes in Computer Science*, vol. 3998, pp. 115–126. Springer (2006). Full version to appear in *Discrete Mathematics*
6. Bermond, J. C., Galtier, J., Klasing, R., Morales, N., Pérennes, S.: Gathering in specific radio networks. In: 8èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, pp. 85–88. Trégastel, France (2006)
7. Bermond, J. C., Galtier, J., Klasing, R., Morales, N., Pérennes, S.: Hardness and approximation of gathering in static radio networks. Parallel Processing Letters **16**(2), 165–183 (2006)
8. Bermond, J. C., Galtier, J., Klasing, R., Morales, N., Pérennes, S.: Hardness and approximation of gathering in static radio networks. In: Proc. Foundation and Algorithms for Wireless Networking, pp. 75–79. Pisa, Italy (2006)
9. Bermond, J. C., Gargano, L., Rescigno, A.: Gathering with minimum delay in tree sensor networks. In: Proceedings of the 15th International Colloquium on Structural Information and Communication Complexity, *Lecture Notes in Computer Science*, vol. 5058, pp. 262–276. Springer (2008)
10. Bermond, J. C., Peters, J.: Efficient gathering in radio grids with interference. In: Septiémes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, pp. 103–106. Presqu'île de Giens, France (2005)
11. Bermond, J. C., Yu, M. L.: Optimal gathering algorithms in multi-hop radio tree networks with interferences. In: Proc. of the Int. Conf. on Ad-Hoc, Mobile, and Wireless Networks, pp. 204–217 (2008)
12. Bertin, P., Bresse, J. F., Sage, B.: Accès haut débit en zone rurale: une solution "ad hoc". France Telecom R & D **22**, 16–18 (2005)
13. Bonifaci, V., Korteweg, P., Marchetti-Spaccamela, A., Stougie, L.: An approximation algorithm for the wireless gathering problem. Operations Research Letters **36**(5), 605–608 (2008)
14. Bonifaci, V., Korteweg, P., Marchetti-Spaccamela, A., Stougie, L.: The distributed wireless gathering problem. In: Proc. Int. Conf. on Algorithmic Aspects in Information and Management, *Lecture Notes in Computer Science*, vol. 5034, pp. 72–83. Springer (2008)
15. Bonifaci, V., Korteweg, P., Marchetti-Spaccamela, A., Stougie, L.: Minimizing average flow time in sensor data gathering. In: Proc. 4th Workshop on Algorithmic Aspects of Wireless Sensor Networks, *Lecture Notes in Computer Science*, vol. 5389, pp. 18–29. Springer (2008)
16. Bonifaci, V., Korteweg, P., Marchetti-Spaccamela, A., Stougie, L.: Minimizing flow time in the wireless gathering problem. In: Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science, pp. 109–120 (2008)
17. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press (1998)
18. Boukerche, A. (ed.): Handbook of Algorithms for Wireless Networking and Mobile Computing. Chapman & Hall (2005)
19. Cidon, I., Kutten, S., Mansour, Y., Peleg, D.: Greedy packet scheduling. SIAM Journal on Computing **24**(1), 148–157 (1995)

20. Coleri, S.: PEDAMACS: Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks. Master's thesis, University of California, Berkeley (2002)
21. Duckworth, W., Manlove, D., Zito, M.: On the approximability of the maximum induced matching problem. Journal of Discrete Algorithms **3**(1), 79–91 (2005)
22. Florens, C., Franceschetti, M., McEliece, R. J.: Lower bounds on data collection time in sensory networks. IEEE Journal on Selected Areas in Communications **22**, 1110– 1120 (2004)
23. Garey, M. R., Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman (1979)
24. Gargano, L.: Time optimal gathering in sensor networks. In: Proceedings of the 14th International Colloquium on Structural Information and Communication Complexity, *Lecture Notes in Computer Science*, vol. 4474, pp. 7–10. Springer (2007)
25. Gargano, L., Rescigno, A. A.: Optimally fast data gathering in sensor networks. In: Proceedings of the 31st Symposium on Mathematical Foundations of Computer Science, *Lecture Notes in Computer Science*, vol. 4162, pp. 399–411. Springer (2006)
26. Hromkovič, J.: Algorithmics for Hard Problems — Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. Springer (2001)
27. Klasing, R., Morales, N., Pérennes, S.: On the complexity of bandwidth allocation in radio networks. Theoretical Computer Science **406**, 225–239 (2008)
28. Korteweg, P.: Online Gathering Algorithms for Wireless Networks. Ph.D. thesis, Eindhoven Technical University (2008)
29. Kumar, V. S. A., Marathe, M. V., Parthasarathy, S., Srinivasan, A.: End-to-end packet-scheduling in wireless ad-hoc networks. In: J. I. Munro (ed.) Proceedings of the 15th Symposium on Discrete Algorithms, pp. 1021–1030 (2004)
30. Morales, N.: Algorithmique de réseaux de communication radio modelisés par de graphes. PhD thesis, Université de Nice-Sophia Antipolis (2007)
31. Papadimitriou, C. H.: Computational Complexity. Addison-Wesley (1994)
32. Perkins, C. E. (ed.): Ad Hoc Networking. Addison-Wesley Professional (2001)
33. Pottie, G. J., Kaiser, W. J.: Wireless integrated network sensors. Communications of the ACM **43**(5), 51–58 (2000)
34. Raghavendra, C. S., Sivalingam, K. M., Znati, T. (eds.): Wireless Sensor Networks. Springer (2004)
35. Revah, Y., Segal, M.: Improved algorithms for data-gathering time in sensor networks II: Ring, tree and grid topologies. In: Proc. of the 3rd IEEE Int. Conf. on Networking and Services (2007)
36. Revah, Y., Segal, M.: Improved lower bounds for data-gathering time in sensor networks. In: Proc. of the 3rd IEEE Int. Conf. on Networking and Services (2007)
37. Schmid, S., Wattenhofer, R.: Algorithmic models for sensor networks. In: Proceedings of the 20th International Parallel and Distributed Processing Symposium (2006)
38. Segal, M., Yedidsion, L.: On real time data-gathering in sensor networks. In: Proc. of the 3rd IEEE Int. Conf. on Mobile, Adhoc and Sensor Systems (2007)
39. Vazirani, V. V.: Approximation Algorithms. Springer, Berlin (2001)
40. Yao, A. C. C.: Probabilistic computations: Towards a unified measure of complexity. In: Proc. of the 18th Symp. on the Foundations of Computer Science, pp. 222–227 (1977)

Chapter 15
Tournament Methods for WLAN: Analysis and Efficiency

Jérôme Galtier

Abstract In the context of radio distributed networks, we present a generalized approach for Medium Access Control (MAC) with a fixed congestion window. Our protocol is quite simple to analyze and can be used in a lot of different situations. We give mathematical evidence showing that our performance is asymptotically tight. We also place ourselves in the WiFi and WiMAX frameworks, and discuss experimental results showing collision reduction of 14% to 21% compared to the best-known methods. We discuss channel capacity improvement and fairness considerations.

Key words: WLAN, MAC protocol, tournament, CSMA, WiFi, WiMAX

15.1 Introduction and Related Works

Radio networks have received in the past few years a growing interest for their ability to offer relatively wide band radio networking. Applications cover a large area of domains, including computer network wireless infrastructures and high speed Internet access for rural areas.

For instance, in WiFi and WiMAX norms, the underlying mechanism [1, 2] (see also [19]) is a 2-layer protocol whose first part relies on a derivative of the Binary Exponential Backoff protocol (BEB). The principle is that when a failure occurs, the transmission protocol delays the following retransmission by some penalty factor. The protocol uses a contention window (CW) mechanism to realize this backoff mechanism. Roughly speaking, the probability of trying an access to the channel is $1/CW$. When a transmission fails, the station increases CW in order to be less demanding for future accesses.

Jérôme Galtier

Orange Labs, 905 Avenue Albert Einstein, 06921 Sophia Antipolis Cedex, France, e-mail: `jerome.galtier@orange-ftgroup.com`

| DIFS | CRP | Packet | SIFS | ACK | DIFS | CRP | Packet | DIFS | CRP | Packet | SIFS | ACK | ... |

Successful transmission   Failed transmission   Successful transmission
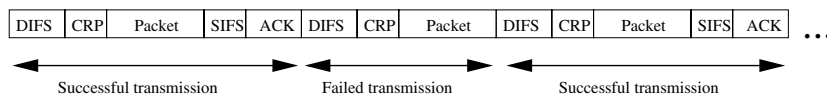
Fig. 15.1 A view of the general frame structure

Already much research work has been done to model the *CW* increase or decrease process. Strong simplifying assumptions are at the basis of some models [8], while other researchers focus on an individual station while considering that the effect of the several contending terminals on the channel can be represented by an occupancy probability p_{occ} (see [5, 22, 23]), following an earlier popular approach on CSMA [4, 18].

In fact, those studies show that on average, the contention window mechanism draws the stations to access the channel with some probability that converges to a value noted $p_{acc}(n)$ which depends on the number of stations simultaneously willing to access the channel. Therefore, several studies have shown that the optimal behavior is when $p_{acc}(n) = \mathcal{O}(1/n)$, and proposed some alternative mechanisms to increase and decrease the contention window in order to reach this value. In [6, 7], the authors aim at guessing the total number of stations trying to emit in order to directly set the value *CW*. In [12], an optimization of the increase or decrease parameters is done to converge to the optimal channel efficiency in terms of capacity. In [13] the authors use the observation of idle slots to deduce the probable number of competing stations. In [14], however, strong evidence is given that, with a dynamic number of stations, all the aforementioned studies should be questioned as the guess for the number of contending stations becomes less and less accurate.

A different branch of CSMA protocols has been initiated by the Hiperlan protocol [20], a twin standard of 802.11a developed in the same period. In this protocol, the contention phase is bounded. The contention phase begins for each terminal with the emission of a burst whose length follows a truncated geometric distribution, and the terminal having the longest burst wins the right to transmit. If several terminals have the same longest burst, this results in a collision. A very similar protocol developed in the context of sensor networks has been called Sift [16].

Another related protocol is the *Contention Tree Algorithm* [9, 10, 17], or CTA for short, also called *Stack Algorithm* [11, 21], which uses a tree to solve contention problems. Although we also use a tree, our protocol is completely different. This algorithm is basically based on feedback, that is, evidence that a collision occured. In the radio context, a feedback is costly since it requires an acknowledgement packet. Instead, we rely on *evidence of occupation*, that is, the fact that a silent terminal can detect that one or more terminals are signaling their presence.

In Figure 15.1 we explain how our protocol, the *selective tournaments*, works. As in the standard 802.11 approach, the transmission begins with a period of sensing after the last emission (either a packet or an ACK). After observing a sufficiently long period with no emission (the DIFS period), the system operates a contention resolution protocol (CRP) that is supposed to select one station, and only one. Then the packet is transmitted. If it is correctly received, the receiving station emits an

ACK after an SIFS period. This is the end of a transmission period and a new transmission period can begin. If no ACK is received back, the new transmission period begins immediately after the failed packet, and the stations start the CRP mechanism just after the DIFS period (note that SIFS < DIFS).

How does the CRP work? In our protocol the time is subdivided into time slots that correspond to rounds of selection. At the beginning, each terminal emits a signal on the first time slot with a certain probability. In the case where the station does not emit, it listens to other signals, and, in the case where it hears at least one other signal, it retires itself from the selection. This process, called *round*, is repeated an appropriate number of times, in order to leave only one remaining station at the end with the highest probability. This method is used in [3].

15.2 Description of the Tournament Method

The present article generalizes this last method, and presents a mathematical framework to analyze its strenghts and weaknesses. As a result, the improved method presents a reduction of collision between 13.9% and 21.1%, resulting in a systematic gain with respect to the original throughput. The gain to the original 802.11b norm is as high as 34.1%, achieving the best-known throughput for this family of protocols. More than that, the new protocol keeps excellent fairness characteristics, as the Jain index indicates.

More than that, we do not use a fixed number of rounds in the tournament as is done in CONTI (in [3] a fixed number of six rounds is assumed). We use a *variable* number of slots of selection, depending on what was previously heard on the channel. We tune our experiments to the case where the amount of contending stations is often between 10 and 100, but our analysis can be very easily extended to any number of contending stations. By adding a sufficient number of minislots, we can reduce the number of collisions to an arbitrarily low level. Since adding a minislot can be statistically more penalizing from the throughput point of view than retransmitting a packet in the case of a collision, a clever balance should be computed to obtain an optimal throughput. In the following, an entry of our protocol will be the variable $E_f(r(1), \ldots, r(k))$, which states whether the collision resolution protocol should be stopped at round k or not.

The key for obtaining good results is in the choice of the probabilities with which a station will decide to keep silent or emit a signal in the CRP phase. Each station takes into account whether signals were emitted or not in the previous rounds to adapt its probability of emission.

In the following, we call try-bit and denote by $r(t)$ the information that at least one station has emitted a signal on the tth round.

In Figure 15.2, we show how the choice of p – the probability that a station emits at a given round of selection – evolves in the course of the rounds of selection and in the function of the previous try-bits chosen by the terminal. The first value (in Figure 15.2, $p = 0.02$) is unique for all the terminals. During the second round, if

the terminal has emitted a signal in the first round (which we denote $r(1) = 1$), the protocol chooses the left part of the tree, and uses $p_1 = 0.33$ for the second round. If on the other hand the terminal did not emit, and did not hear any signal from other terminals ($r(1) = 0$), then the protocol chooses the right part of the tree and uses $p_0 = 0.12$ for the second round. In the case where the terminal did not emit and actually hears a signal from another station, it retires and leaves to other stations the right to send the following packet. As a result, the probability in the second round is necessarily $p_{r(1)}$. The realization of the second round will determine the value of $r(2)$, and the third round will be governed by the probability $p_{r(1)r(2)}$ in the tree of Figure 15.2. We plot the whole process in Figure 15.3. Note that each station has a local try-bit $R(t)$ at round t which equals $r(t)$ as long as the station is not eliminated. In Figure 15.3, we also make use of the pf and Ef functions. pf is a function that, given the values $R(1) \ldots R(t-1)$ of the previous rounds of the tournament, outputs a probability $P[R(t)]$ of emission of a signal during the actual round. In the following, we will note $w = \text{``}R(1)\text{''} \ldots \text{``}R(t)\text{''}$ as the word containing the history of the tournament and use the notation p_w to model what will be $pf[R(1), \ldots, R(t-1)]$. Also Ef is a function of $R(1) \ldots R(t)$ which indicates to the terminal that in the case of success for round t, it has to proceed to emit the packet ($Ef = yes$) or start another round of selection ($Ef = no$). Note also that in the bottom-left part of Figure 15.2, only three rounds of elimination are performed, while four are requested in the other parts of the tree. Depending on the previous states, that is, on $r(1), r(2), \ldots, r(t)$, our protocol can decide to perform the transmission. In the case of Figure 15.2, until the third round, three successive signals have been emitted, so the system forsees that one station is attempting to communicate with high priority, and proceeds to transmission.

We manage to find a tight approximation of the behavior of this protocol when the number of rounds increases. More precisely, if we denote by q_n the probability that n stations try to emit in the system, and if we introduce the function
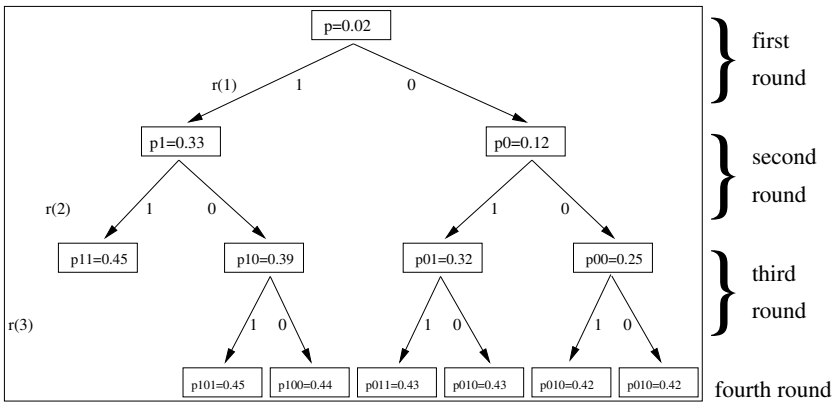
Fig. 15.2 Choice of the values of p in the course of the rounds of selection
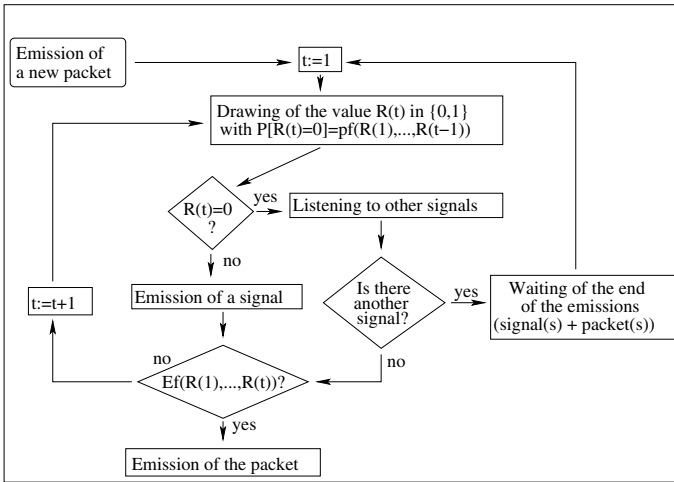
Fig. 15.3 Global diagram for the rounds of selection in a terminal

$$f(x) = \sum_{n \geq 1} q_n x^n, \tag{15.1}$$

then we can show that, tuning the p coefficients to this probability space, the rate of collision $1 - \rho$ observed if the protocols stops after k rounds will be fairly approximated by

$$1 - \rho \approx \frac{\left(\int_0^1 \sqrt{f''(t)} dt \right)^2}{2^{k+1}}.$$

The article is organized as follows. A mathematical modeling in the next section investigates analytically the optimization issues raised by the problem of the choice of the p's, and gives some tight bounds for this question. The reader that desires to know the protocol without the mathematics can skip this section. A practical implementation of the mathematical ideas, allowing the computation of the values of the p's, is given in Section 15.3, which gives a new protocol for the WiFi/WiMAX networks. Here again, the reader will not need to program the whole framework to implement and reproduce our protocol, which is described in terms of parameters in Table 15.2. Finally, our protocol is compared with other ones in Section 15.4, where some numerical results are presented. A summary of the notations used throughout the chapter is given in Table 15.1.

15.3 Mathematical Analysis

During the mathematical part of the analysis, we study the case where the CRP stops after k rounds. In the following we denote by m the number 2^k.

Table 15.1 Notations used in this chapter

k	Number of rounds of selection.	z_i	Riemann steps for f' in $[0,1]$, $i \in \{0,\ldots,m\}$.
m	2^k.	φ	Continuous function in $[0,1]$ to be approxi -mated.
$r(t)$	Try-bit at the tth round of selection, $t \in \{1,\ldots,k\}$.	ψ	Piecewise constant function that approximate φ.
$R(t)$	Local try-bit (at a station) at the tth round of selection, $t \in \{1,\ldots,k\}$.	c_m	Best approximation gap for the approximation of φ with m pieces.
q_n	Probability that n station try to emit.	σ	Increasing function from $\mathbb{N}$ to $\mathbb{N}$ that emphasizes extraction.
w	Word in the alphabet $\{0,1\}$.	A	Function of $\mathbb{R}^{m+1}$ to $\mathbb{R}$ that reaches approxi-mation for φ.
$l(w)$	Length of the word w.		
$\#(w)$	Binary value represented by w.	$\hbar$	Density step function defined after $z_.$, see equation (15.6).
p	Probability that a station emits a sig-nal at the first round of selection.	h^*	Density function that minimizes $\int_0^1 f''(t)/h(t)dt$, see Proposition 15.2.
p_w	Probability that a non-eliminated sta-tion emits a signal at the round $l(w)+1$, given that the preceding try-bits where $r(1),\ldots,r(l(w))=w$.	$\hat{h}$	Density function taken for the algorithmic choices of $z_.$, $\hat{h}(x)=\sqrt{f''(x)}$.
f	Generating function of $q_.$, see equation (15.1).	M	Large number compared to m.
		L	Level variable ($L=2^{k-l(w)-1}$).
f_w	Generating function of the number of non-eliminated stations, in the event w.	N	Maximum number of foreseen stations.
g	Generating function of the number of non-eliminated stations after k rounds.	α	Parameter to set the values of $q_.$, see equa-tion (15.13).
ρ	Success rate (as opposed to the colli-sion rate).	x_i	In the experiments, amount of packets that an individual station has emitted.
δ_w	Local step, see equation (15.2).	π_w	In the experiments, the probability that the se-quence w will be used for signaling.
y_w	Cumulative step, see equation (15.3).		

We denote by q_n the probability that n stations try to emit in the system. We have necessarily $q_n \geq 0$ and $\sum_{n\geq 1} q_n = 1$. We introduce f, which in the following we will call the generating function of the distribution of stations, defined by:

$$f(x) = \sum_{n\geq 1} q_n x^n.$$

We try now to characterize the distribution after the transmission on the first mini-slot. Let f_1 and f_0 be generating functions for the number of stations still in contention depending, respectively, on whether or not there was a transmission in the previous slot. If every station emits a signal with probability p, then the probability that n stations emit is given by

$$P[n \text{ stations emit a signal}] = \sum_{i\geq n} \binom{i}{n} q_i p^n (1-p)^{i-n}.$$

Therefore, the distribution of the number of stations that emit is characterized by a function f_1 analog of f, defined by

$$f_1(x) = \sum_{i\geq 1} P[i \text{ stations emit a signal}] x^i,$$

and we deduce logically that

$$f_1(x) = \sum_{n \geq 1} \sum_{1 \leq i \leq n} \binom{n}{i} q_n p^i (1-p)^{n-i} x^i$$
$$= \sum_{n \geq 1} q_n [(px+1-p)^n - (1-p)^n]$$
$$= f(px+1-p) - f(1-p).$$

Similarly, in the event where no signal has been emitted in the first round, we can deduce some information on the distribution of the number of stations. Indeed, the probability that n stations remain silent is $(1-p)^n$. Therefore, if we write

$$f_0(x) = \sum_{i \geq 0} P[i \text{ stations are present and remain silent}] x^i$$

then we obtain

$$f_0(x) = \sum_{i \geq 0} q_i x^i (1-p)^i = f((1-p)x).$$

And we see that at the end of the first round of selection, the distribution of the whole set of surviving station can be known by the mathematical function $f_0 + f_1$. We can also note that the distribution in the case where the event $r(0)$ occurs (either 0 or 1) is given by $f_{r(0)}/f_{r(0)}(1)$.

By extension, if we let w be a word in the alphabet $\{0,1\}$, and $w0$ (or $w1$) the same word to which the letter "0" (or "1") is added, and if we let p_w and f_w, respectively, be the probability and generating function corresponding to step w, then (setting $f_\emptyset = f$) the following inductive formulas hold:

$$\begin{cases} f_{w1}(x) &= f_w(p_w x + 1 - p_w) - f_w(1 - p_w) \\ f_{w0}(x) &= f_w((1 - p_w)x) \end{cases}$$

We observe that the probability of the event of the choice $w = r(1) \ldots r(t)$ is $f_w(1)$. Given that the event w occurs, the distribution of the number of stations is characterized by $f_{r(1)\ldots r(k)}/f_{r(1)\ldots r(k)}(1)$. If we denote by $l(w)$ the length of the word w, then the global distribution g for all the event space after k rounds of selection is given by the sum of all the f_w's that correspond to an event after k rounds (this is true if and only if $l(w) = k$, since $E_f(w)$ is true when $l(w)$ reaches k), and therefore,

$$g(x) = \sum_{w: l(w)=k} f_w(x).$$

The probability of success ρ of the rounds of selection is the probability that only one station remains. It is given by the first term of the integral series of g, that is, $g'(0)$. Therefore,

$$\rho = \sum_{w: l(w)=k} f'_w(0).$$

In the following we evaluate the value of $f'_w(0)$. We therefore denote, for any word w in the alphabet $\{0,1\}$, the quantity defined inductively by $\delta_\emptyset = 1$ and

$$\begin{cases} \delta_{w0} &= (1 - p_w)\delta_w \\ \delta_{w1} &= p_w \delta_w \end{cases} \tag{15.2}$$

Then we note $w_a < w_b$ if their corresponding binary values verify inequality (15.2), and, using the convention $y_\emptyset = 0$, we set

$$y_w = \sum_{v<w:l(v)=l(w)} \delta_v. \tag{15.3}$$

Lemma 15.1. $y_{w0} = y_w$.

Proof. It is easy to see that $\delta_{w1} + \delta_{w0} = \delta_w$. Therefore,

$$y_{w0} = \sum_{v<w0:l(v)=l(w0)} \delta_v = \sum_{u<w:l(u)=l(w)} (\delta_{u0} + \delta_{u1}) = \sum_{u<w:l(u)=l(w)} \delta_u = y_w \qquad \square$$

Lemma 15.2. $f'_w(x) = \delta_w f'(y_w + \delta_w x)$ *for all* $x \in [0,1]$.

Proof. Obviously $\delta_\emptyset f'(y_\emptyset + \delta_\emptyset x) = f'(x) = f'_\emptyset$. Then we apply another induction on w. We suppose that the statement is established for w and show that it is true for $w0$ and $w1$. Indeed

$$\begin{aligned}
f'_{w0}(x) &= (1-p_w)f'_w((1-p_w)x) \\
&= (1-p_w)\delta_w f'(y_w + \delta_w(1-p_w)x) \\
&= \delta_{w0} f'(y_{w0} + \delta_{w0}x);
\end{aligned}$$

moreover, noticing from equation (15.3) that $y_{w1} = y_{w0} + \delta_{w0}$, we have,

$$\begin{aligned}
f'_{w1}(x) &= p_w f'_w(p_w x + 1 - p_w), \\
&= p_w \delta_w f'(y_w + \delta_w(p_w x + 1 - p_w)), \\
&= \delta_{w1} f'(y_{w0} + \delta_{w0} + \delta_{w1}x), \\
&= \delta_{w1} f'(y_{w1} + \delta_{w1}x).
\end{aligned}$$

$\square$

And therefore

$$\rho = \sum_{w:l(w)=k} \delta_w f'(y_w).$$

This formula exactly says that we aim at approximating the integral of f' by a Riemann integral. In other words, if we are given $m - 1 = 2^k - 1$ real numbers $z_1, \ldots, z_{m-1}$ in $(0,1)$, with $0 = z_0 < z_1 < \cdots < z_{m-1} < z_m = 1$, then the quantity

$$\rho = \sum_{i \in \{1,\ldots,m\}} (z_i - z_{i-1})f'(z_{i-1})$$

is the approximation of the integral of f' by a piecewise constant function having m steps. This fact is illustrated by Figure 15.4. In this figure we draw the f' function, of which the integral between 0 and 1 exactly equals 1 (since $f(1) - f(0) = 1$). Points have been chosen to approximate this integral by a lower bound piecewise constant function. The rate of collision of our protocol will be exactly equal to the area in gray in Figure 15.4, which is also the approximation default. Therefore, if we have
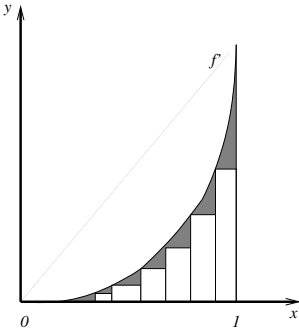
Fig. 15.4 Interpretation of the collision rate in terms of Riemann integral

the best values of $z_0, \ldots, z_m$ for this integral, it is sufficient to set $y_w = z_{\#(w)}$, where $\#(w)$ is the numerical binary value that is represented by w. The reader can verify that this is obtained by setting $L = 2^{k-lw-1}$ and

$$p_w = \frac{z_{L(2\#(w)+2)} - z_{L(2\#(w)+1)}}{z_{L(2\#(w)+2)} - z_{L2\#(w)}}.$$

Theorem 15.1. *For all protocols of selection governed by a series of selective rounds as indicated in Figure 15.2, we can associate a series of $m+1$ real numbers $z_0, \ldots, z_m$ in $[0,1]$ with $0 = z_0 < z_1 < \cdots < z_{m-1} < z_m = 1$, such that the probability of success (non-collision) of the protocol is given by*

$$\rho = \sum_{i\in\{1,\ldots,m\}} (z_i - z_{i-1})f'(z_{i-1}). \tag{15.4}$$

In this case, the probabilities chosen to operate the different rounds of selection are given by $L = 2^{k-lw-1}$ and

$$p_w = \frac{z_{L(2\#(w)+2)} - z_{L(2\#(w)+1)}}{z_{L(2\#(w)+2)} - z_{L2\#(w)}}. \tag{15.5}$$

Conversely, with every family of real numbers $z_0, \ldots, z_m$ verifying

$$0 = z_0 < z_1 < \cdots < z_{m-1} < z_m = 1,$$

we can associate a protocol of selection whose probability of success and probabilities are given by equations (15.4) and (15.5).

So we are now left with the problem of finding optimal values for z. Analyzing a little further the value of ρ, we obtain the formula

$$\begin{aligned}
1 - \rho &= \int_0^1 f'(t)dt - \sum_{i\in\{1,\ldots,m\}} (z_i - z_{i-1})f'(z_{i-1}) \\
&= \sum_{i\in\{1,\ldots,m\}} f(z_i) - f(z_{i-1}) - (z_i - z_{i-1})f'(z_{i-1}).
\end{aligned}$$

Three lemmas will allow us to analyze it.

Lemma 15.3. *Let $\hbar$ be the piecewise constant function defined by*

$$\hbar : x \mapsto \frac{1}{m(z_i - z_{i-1})} \ \text{for } x \in [z_{i-1}; z_i[. \tag{15.6}$$

We have $\int_1^0 \hbar(t)dt = 1$ and

$$\frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt = \sum_{i \in \{1,\dots,m\}} \int_{z_{i-1}}^{z_i} \frac{z_i - z_{i-1}}{2} f''(t)dt$$

$$= \sum_{i \in \{1,\dots,m\}} \frac{(z_i - z_{i-1})(f'(z_i) - f'(z_{i-1}))}{2}.$$

Moreover,

$$1 - \rho - \frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt \geq -\frac{1}{12} \sum_{i \in \{1,\dots,m\}} (z_i - z_{i-1})^3 f'''(z_i) \tag{15.7}$$

and

$$1 - \rho - \frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt \leq -\frac{1}{12} \sum_{i \in \{1,\dots,m\}} (z_i - z_{i-1})^3 f'''(z_{i-1}). \tag{15.8}$$

Proof.

$$1 - \rho - \frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt$$

$$= \sum_{i \in \{1,\dots,m\}} f(z_i) - f(z_{i-1}) - (z_i - z_{i-1})f'(z_{i-1}) - \frac{(z_i - z_{i-1})(f'(z_i) - f'(z_{i-1}))}{2}$$

$$= \sum_{i \in \{1,\dots,m\}} f(z_i) - f(z_{i-1}) - (z_i - z_{i-1})\frac{f'(z_i) + f'(z_{i-1})}{2}.$$

But f is a harmonic function with positive coefficients, and with radius of convergence at least 1; therefore,

$$f(z_i) = \sum_{j \geq 0} \frac{(z_i - z_{i-1})^j}{j!} f^{(j)}(z_{i-1})$$

$$f'(z_i) = \sum_{j \geq 1} \frac{(z_i - z_{i-1})^{j-1}}{(j-1)!} f^{(j)}(z_{i-1})$$

and we have

$$f(z_i) - f(z_{i-1}) - \frac{z_i - z_{i-1}}{2}(f'(z_i) + f'(z_{i-1}))$$

$$= (z_i - z_{i-1})^3 \sum_{j \geq 3} (z_i - z_{i-1})^{j-3} \left[\frac{1}{j!} - \frac{1}{2(j-1)!} \right] f^{(j)}(z_{i-1}),$$

$$= -(z_i - z_{i-1})^3 \sum_{j \geq 3} (z_i - z_{i-1})^{j-3} \frac{1}{(j-1)!} \left[\frac{1}{2} - \frac{1}{j} \right] f^{(j)}(z_{i-1}).$$

We note that all the derivatives of f are positive, and therefore all the terms of this series are non-positive. Hence, the inequality (15.8). Moreover,

$$\sum_{j \geq 3} (z_i - z_{i-1})^{j-3} \left[\frac{1}{2(j-1)!} - \frac{1}{j!} \right] f^{(j)}(z_{i-1})$$

$$= \sum_{j \geq 3} (z_i - z_{i-1})^{j-3} \frac{j-2}{2(j!)} f^{(j)}(z_{i-1})$$

$$\leq \sum_{j \geq 3} (z_i - z_{i-1})^{j-3} \frac{1}{12(j-3)!} f^{(j)}(z_{i-1})$$

$$\leq \frac{1}{12} f'''(z_i)$$

$\square$

Lemma 15.4. *Suppose $f''(0) > 0$. Let the real numbers $z_0, \ldots, z_m$ in $[0,1]$, with $0 = z_0 < z_1 < \cdots < z_{m-1} < z_m = 1$, achieve the maximum value of $\rho = \sum_{i \in \{1,\ldots,m\}} (z_i - z_{i-1}) f'(z_{i-1})$. Then we have*

$$\sum_{i \in \{1,\ldots,m\}} (z_i - z_{i-1})^2 \leq \frac{2}{m f''(0)}, \tag{15.9}$$

and

$$z_i - z_{i-1} \leq \sqrt{\frac{2}{f''(0)}} \frac{1}{\sqrt{m}} \qquad \forall i \in \{1,\ldots,m\}. \tag{15.10}$$

Proof. We have

$$1 - \rho = \sum_{i \in \{1,\ldots,m\}} f(z_i) - f(z_{i-1}) - (z_i - z_{i-1}) f'(z_{i-1}),$$

and by Taylor expansion, there exists b_{i-1} in the interval (z_{i-1}, z_i) such that

$$f(z_i) - f(z_{i-1}) = (z_i - z_{i-1}) f'(z_{i-1}) + \frac{(z_i - z_{i-1})^2}{2} f''(b_{i-1}).$$

Therefore, $1 - \rho = \displaystyle\sum_{i \in \{1,\ldots,m\}} \frac{(z_i - z_{i-1})^2}{2} f''(b_{i-1})$. Since ρ is a maximum value for all the choices of z_i, necessarily

$$1 - \rho \leq 1/m \tag{15.11}$$

(indeed, if we take $z_i = i/m$, we obtain a solution verifying $1 - \rho \leq 1/m$). Therefore, $z_i - z_{i-1}$ tends to 0 when m tends to infinity. The inequality (15.11) taken term by term gives $z_i - z_{i-1} \leq \sqrt{\frac{2}{f''(0)} \frac{1}{\sqrt{m}}}$, and since all the $f''(b_i)$ are bounded below by $f''(0)$, we obtain

$$\sum_{i \in \{1,\dots,m\}} (z_i - z_{i-1})^2 \leq \frac{2}{mf''(0)}.$$

$\square$

Lemma 15.5. *If f'' is piecewise continuous, the minimum of the value $\int_0^1 \frac{f''(t)}{h(t)} dt$ on the functions h piecewise continuous and verifying $\int_0^1 h(t)dt = 1$ is obtained by $h^* : x \mapsto \frac{\sqrt{f''(x)}}{\int_0^1 \sqrt{f''(t)}dt}$, and therefore equals $\left(\int_0^1 \sqrt{f''(t)}dt\right)^2$.*

The proof of Lemma 15.5 is easily obtained if f'' is a piecewise constant function, and we use uniform convergence of piecewise constant functions to piecewise continuous functions.

Theorem 15.2. *If $f''(0) > 0$, whatever the series of m values of z used, we have*

$$1 - \rho \geq \frac{\left(\int_0^1 \sqrt{f''(t)}dt\right)^2}{2m} - \frac{f''(1)}{3\sqrt{2}f''(0)^{3/2}} \frac{1}{m^{3/2}}.$$

Proof. By Lemma (15.3) we have:

$$1 - \rho \geq \frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt - \frac{1}{12} \sum_{i \in \{1,\dots,m\}} (z_i - z_{i-1})^3 f'''(z_i).$$

Then, applying Lemma (15.5) for $\hbar = h$ gives

$$1 - \rho \geq \frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt - \frac{1}{12} \sum_{i \in \{1,\dots,m\}} (z_i - z_{i-1})^3 f'''(z_i).$$

Then, the inequality (15.10) of Lemma (15.4) gives

$$1 - \rho \geq \frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt - \frac{1}{12\sqrt{m}} \sqrt{\frac{2}{f''(0)}} \sum_{i \in \{1,\dots,m\}} (z_i - z_{i-1})^2 f'''(z_i).$$

Since $f'''(z_i) \leq f'''(0)$, it gives

$$1 - \rho \geq \frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt - \frac{f'''(0)}{12\sqrt{m}} \sqrt{\frac{2}{f''(0)}} \sum_{i \in \{1,\dots,m\}} (z_i - z_{i-1})^2.$$

Finally, the inequality (15.9) of Lemma (15.4) gives

$$1 - \rho \geq \frac{1}{2m} \int_0^1 \frac{f''(t)}{\hbar(t)} dt - \frac{f'''(0)}{12\sqrt{m}} \sqrt{\frac{2}{f''(0)}} \frac{2}{mf''(0)}.$$

Hence the result.

□

This result clearly shows the limit in efficiency of the protocol whatever the chosen values of probabilities, and therefore the maximum we can asymptotically expect, which is

$$1 - \rho \approx \frac{\left(\int_0^1 \sqrt{f''(t)} dt \right)^2}{2m}.$$

The strategy of approaching h by a piecewise constant function therefore gives a result asymptotically optimal.

We note also that as the number of rounds increases, the collision rate systematically decreases, and this factor of division converges to 2 as the number of rounds increases. It means that we can reduce the collision rate to an arbitrary low level by using a reasonably small number of additional rounds. For applications that suffer from retransmissions (and jitter) this property can have considerable impact.

However, further experiments showed that in the 802.11b framework, where the jitter is not important and the throughput is to be optimized, a good number of minislots to use is between six and eight.

Some more results on the approximation of a function by Riemann integrals are worth noting. We are interested in the following problem:

15.1. Let φ be a non-decreasing function from $[0,1]$ to $[0,1]$, with $\varphi(0) = 0$ and $\varphi(1) = 1$. We are looking for a piecewise constant function ψ, with $m+1$ pieces, such that $\psi(z) \leq \varphi(z)$ for all $z \in [0,1]$ and

$$\int_0^1 (\varphi(z) - \psi(z)) \, dz$$

is minimum.

Clearly, this problem is the optimization formulation of the preceding issue when, for $z \in [0,1]$, $\varphi(z) = f'(z)/f'(1)$.

Theorem 15.3. *If $z \mapsto z\varphi(z)$ is convex, $m = 1$, and φ is continuously derivable, then the minimum for g is reached with a step z verifying $(1-z)\varphi'(z) = \varphi(z)$.*

Proof. If the step is $z \in [0,1]$, then

$$\int_0^1 (\varphi(t) - \psi(t)) dt = \int_0^1 \varphi(t) dt - (1-z)\varphi(z).$$

Note that necessarily there is some $z^* \in (0,1)$ such that $\varphi(z^*) > 0$, and this particular z^* does better than $z = 0$ or $z = 1$. The best z necessarily then verifies $\frac{d}{dz}(1-z)\varphi(z) = 0$. □

Theorem 15.4. *For a fixed m, for any φ function, there is a ψ function that reaches the minimum. This minimum will be noted c_m.*

Proof. Set

$$c_m = \inf\left\{ \int_0^1 (\varphi(t) - \psi(t))dt : \psi \text{ piecewise constant with } m+1 \text{ pieces, and } \psi \le \varphi \right\}.$$

Let ψ_p be a series of piecewise constant functions, with $m+1$ pieces, such that

$$\lim_{p\to\infty} \int_O^1 (\varphi(t) - \psi_p(t))dt = c_m.$$

Let $z_p^{(1)} \le \cdots \le z_p^{(m)}$ be the points of non-constancy of ψ_p. Since $[0,1]$ is compact, let us extract a converging series $z_{\sigma_1(p)}^{(1)}$ (that is, σ_1 is an increasing function from $\mathbb{N}$ to $\mathbb{N}$ such that the series $z_{\sigma_1(p)}^{(1)}$, $p \in \mathbb{N}$, is converging), and σ_2 such that $z_{\sigma_2(\sigma_1(p))}^{(2)}$ is converging, and so on until σ_m such that $z_{\sigma_m o \ldots o \sigma_1(p)}^{(m)}$ is converging. Setting $\sigma = \sigma_m \circ \cdots \circ \sigma_1$ we have that σ is increasing from $\mathbb{N}$ to $\mathbb{N}$, and for each $i \in \{1, \ldots, m\}$, $z_{\sigma(p)}^{(i)}$, $p \in \mathbb{N}$ is converging.

Let us then note $z_*^{(i)} = \lim_{p\to\infty} z_{\sigma(p)}^{(i)}$, for $i \in \{1, \ldots, m\}$, $z_*^{(0)} = 0$, $z_*^{(m+1)} = 1$, and ψ^* such that for $i \in \{1, \ldots, m+1\}$, and $z \in [z_*^{(i-1)}, z_*^{(i)})$, $\psi^*(z) = \varphi(z_*^{(i-1)})$.

Let $\varepsilon > 0$ be a real number. There is an $\eta > 0$ such that for all $i \in \{1, \ldots, m\}$, $|z - z_*^{(i)}| < \eta$ implies $|\varphi(z) - \varphi(z_*^{(i)})| < \varepsilon$. If $\eta > \varepsilon$, set $\eta = \varepsilon$. Then there is a $P \in \mathbb{N}$ such that $p \ge P$ implies $|z_*^{(i)} - z_{\sigma(p)}^{(i)}| \le \eta$.

We see that for each $i \in \{1, \ldots, m\}$,

$$\psi_*(z_*^{(i)}) = \varphi(z_*^{(i)}) \ge \varphi(z_p^{(i)}) - \varepsilon = \psi_p(z_p^{(i)}) - \varepsilon$$

for $p \ge P$, and therefore

$$\int_0^1 (\psi_*(t) - \psi_p(t))\,dt \ge -\varepsilon - m\eta \ge -(m+1)\varepsilon.$$

This is true for all $\varepsilon > 0$, and so $\int_0^1 \psi_*(t)dt \ge c_m$. □

Theorem 15.5. *Let A be the function from $\mathbb{R}^m$ to $\mathbb{R}$ given by*

$$A(z_1, \ldots, z_m) = \sum_{i=2}^{i=m} (z_i - z_{i-1})\varphi(z_{i-1}) + (1 - z_m)\varphi(z_m).$$

Then

$$\max_{(z_1, \ldots, z_m) \in [0,1]^m} A(z_1, \ldots, z_m) = c_m.$$

Proof. It suffices to show that for the maximum we have $z_1 \le \cdots \le z_m$. Note that if $\alpha < \beta$, then

$$(\beta - \alpha)\varphi(\alpha) + (1 - \beta)\varphi(\beta) > (\alpha - \beta)\varphi(\beta) + (1 - \alpha)\varphi(\alpha).$$

It follows that

$$A(z_1, \ldots, z_{i-1}, \alpha, \beta, z_{i+1}, \ldots, z_m) \ge A(z_1, \ldots, z_{i-1}, \beta, \alpha, z_{i+1}, \ldots, z_m).$$

Therefore, ordering the arguments of A maximizes its results (use, for instance, bubble sort). □

Those results open tools for promising optimization.

15.4 Practical Implementation

The basic principles of an optimization based on our mathematical analysis are as follows. A preliminary step consists in fixing a scenario, that is, the probabilities that a given number of stations appears. For instance, we set, as previously,

$$P[\text{Number of emitting stations} = n] = q_n.$$

15.4.1 Setting the Approximation Points

We consider

$$f(x) = \sum_{n \geq 1} q_n x^n.$$

Then we have an $\hat{h}$ function defined by $\hat{h}(x) = \sqrt{f''(x)}$ ($\hat{h}$ is the equivalent of h^* in the previous section, but without the normalization $\int_0^1 h^*(t)dt = 1$). And we take a number M largely greater than $m = 2^k$. We compute H as follows

$$H(0) \quad = 0$$

$$H(i+1) = H(i) + \hat{h}\left(\frac{i+1/2}{M}\right).$$

We define z_j for $j \in \{0, \ldots, m\}$ by

$$z_j = \begin{cases} 0, & \text{if } j = 0 \\ \frac{1}{M} \min\left\{ i : \frac{H(i)}{H(M-1)} \geq \frac{j}{m} \right\} & \text{if } j \in \{1, \ldots, m-1\}, \\ 1 & \text{if } j = m. \end{cases}$$

Finally, we set $L = 2^{k-lw-1}$ and

$$p_w = \frac{z_{L(2\#(w)+2)} - z_{L(2\#(w)+1)}}{z_{L(2\#(w)+2)} - z_{L2\#(w)}}, \tag{15.12}$$

where w is a word in the $\{0, 1\}$ alphabet, $\#(w)$ represents the numerical binary value denoted by w, and $l(w)$ is the length of the word w.

15.4.2 Defining Varying Number of Rounds

In order to know whether a supplementary round of CRP would be necessary, it is necessary to take into account the time slot interval for the CRP (which turns out to be $\tau_{sifs} = 20\mu s$ in practice, following 802.11b standards) and the entire time devoted to a collision ($\tau_c = 1371\mu s$ taking 802.11b at 11 Mbit/s and packets of 1,500 bytes).

We need to compare, for some w with $l(w) < k - 1$, the probability of loss in two cases:

- the loss of having an additional minislot after word w, that is, with $L = 2^{k-lw-1}$,

$$l^w_{sifs} = \tau_{sifs}(f(z_{L(2\#(w)+1)}) - f(z_{L(2\#(w)+2)}))$$

- the loss due to the possible collision if we decide instead to drop the round of selection,

$$l^w_c = \tau_c(f'(z_{L(2\#(w)+1)}) - f'(z_{L(2\#(w))}))(z_{L(2\#(w)+2)} - z_{L(2\#(w)+1)}).$$

The round will be performed only if $l^w_c > l^w_{sifs}$; otherwise, the rounds after w (that is, $w0$, $w1$, and the possible following ones) will not be performed. Note also that if, further, for a prefix of w noted v we have $l^v_c \leq l^v_{sifs}$, the rounds after v (including w) will not be performed.

15.5 Numerical Results

In a first part of this section, we optimize our throughput, in order to obtain some fixed values for the p_w's. Then we compare our protocol to a set of other ones. Along with the native 802.11b protocol [1], we compare it to three high-performing protocols, the Idle Sense protocol [13], the additive congestion window increase/decrease protocol [12], and CONTI [3]. Finally we concentrate on fairness issues for these different schemes, based on the Jain index [15].

15.5.1 Tuning of the Probabilities

An essential step is to set the values of q_n. One idea is to set a preferred interval of operation, say $\{2, \ldots, N\}$, and fix, for $n \in \{2, \ldots, N\}$ and some $\alpha \in [0, 1]$,

$$q_n = \frac{n^{-\alpha}}{\sum_{i=2}^{i=N} i^{-\alpha}}. \tag{15.13}$$

This distribution allows us to take into account in a balanced way loaded or non-loaded networks. In Figure 15.5, we show different curves of throughput obtained for $N = 100$ and various values of α.

Experts in 802.11b can notice that throughputs achieved by these functions are performing remarkably well with respect to the other protocols. The value $\alpha = 0$ allows us to give equal weights to all the events. In practice, we see that this global optimization tends to pay more attention to the cases where more stations are present (50 to 100) at the price of more collisions when two to five stations are present in the system. In contrast, $\alpha = 1$ performs well when a small number of stations are present at the price of worse performance for over 60 stations. A good value seems to be
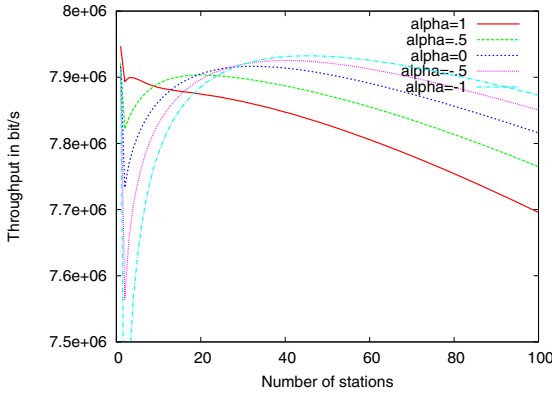
Fig. 15.5 Various throughput obtained playing with α

$\alpha = .7$, where the troughput varies between 8.0 and 7.6 Mbit/s in Figure 15.5. In the following we will set $\alpha = .7$. For the sake of completeness, we give in Table 15.2 our probability values so that the reader can replicate our experiments without further consideration of the choice of α. Note that in Table 15.2, the probabilities π of issuing a sequence are given. Such probabilities can be obtained by

$$\pi_{r_0 r_1 \ldots r_t} = p \; p_{r_0} \; p_{r_0 r_1} \; \cdots \; p_{r_0 r_1 \ldots r_t}$$

or equivalently, from equation (15.12),

$$\pi_w = z_{L(2\#(w)+2)} - z_{L(2\#(w)+1)}.$$

The value of π_{111111} does not appear, which means that if the protocol experiences five times a minislot of signaling, it immediately sends the packet, without performing a sixth round. We note that a value appears for $\pi_{1010000}$. It means that after the seventh minislot of selection, if the sequence "101000" appears, then another set of selection will be performed. A last remark is that the "last positive signal" is replaced by the direct sending of the packet. Hence, instead of signaling "00001", a terminal will signal "0000" and then will proceed to send.

This behavior is indeed in agreement with what was expected. When the first rounds of selections show "a lot of zeros" (the order in which the zeros appear is of course important), the system evaluates that fewer stations are present than were expected, and tries more rapidly to send the packet. If conversely, the first rounds show that a lot of stations are competing, then more rounds of selection are performed. As a consequence, the total number of rounds adapts between six and eight.

Table 15.2 Values obtained for the π.s for $\alpha = .7$ and $N = 100$

$\pi_{11111} = .0009040$	$\pi_{1001110} = .0013427$	$\pi_{0100110} = .0068607$	$\pi_{00100100} = .0094375$	
$\pi_{111110} = .0009326$	$\pi_{100110} = .0013847$	$\pi_{0100101} = .0035533$	$\pi_{0010001} = .0096912$	
$\pi_{11110} = .0009632$	$\pi_{1001100} = .0014324$	$\pi_{01001010} = .0036392$	$\pi_{00100010} = .0099525$	
$\pi_{111100} = .0009956$	$\pi_{100101} = .0014820$	$\pi_{0100100} = .0037250$	$\pi_{0010000} = .0102233$	
$\pi_{11101} = .0010299$	$\pi_{1001010} = .0015316$	$\pi_{01001000} = .0038146$	$\pi_{00100000} = .0104999$	
$\pi_{111010} = .0010681$	$\pi_{100100} = .0015869$	$\pi_{0100011} = .0039081$	$\pi_{0001111} = .0107879$	
$\pi_{11100} = .0011043$	$\pi_{1001000} = .0016441$	$\pi_{01000110} = .0040016$	$\pi_{00011110} = .0110836$	
$\pi_{111000} = .0011463$	$\pi_{100011} = .0017032$	$\pi_{0100010} = .0041007$	$\pi_{0001110} = .0113868$	
$\pi_{11011} = .0011901$	$\pi_{1000110} = .0017662$	$\pi_{01000100} = .0041999$	$\pi_{00011100} = .0117015$	
$\pi_{110110} = .0006122$	$\pi_{100010} = .0018310$	$\pi_{0100001} = .0043010$	$\pi_{0001101} = .0120239$	
$\pi_{1101100} = .0006256$	$\pi_{1000100} = .0019016$	$\pi_{01000010} = .0044078$	$\pi_{00011010} = .0123558$	
$\pi_{110101} = .0006370$	$\pi_{100001} = .0019721$	$\pi_{0100000} = .0045166$	$\pi_{0001100} = .0127010$	
$\pi_{1101010} = .0006504$	$\pi_{1000010} = .0020503$	$\pi_{01000000} = .0046272$	$\pi_{00011000} = .0130558$	
$\pi_{110100} = .0006637$	$\pi_{1000000} = .0022163$	$\pi_{0011111} = .0047416$	$\pi_{0001011} = .0134201$	
$\pi_{1101000} = .0006790$	$\pi_{011111} = .0023040$	$\pi_{00111110} = .0048599$	$\pi_{00010110} = .0137977$	
$\pi_{110011} = .0006923$	$\pi_{0111110} = .0023994$	$\pi_{0011110} = .0049800$	$\pi_{0001010} = .0141849$	
$\pi_{1100110} = .0007076$	$\pi_{011110} = .0024967$	$\pi_{00111100} = .0051040$	$\pi_{00010100} = .0145854$	
$\pi_{110010} = .0007228$	$\pi_{0111100} = .0025997$	$\pi_{0011101} = .0052318$	$\pi_{0001001} = .0149993$	
$\pi_{1100100} = .0007400$	$\pi_{011101} = .0027103$	$\pi_{00111010} = .0053634$	$\pi_{00010010} = .0154266$	
$\pi_{110001} = .0007572$	$\pi_{0111010} = .0028228$	$\pi_{0011100} = .0054988$	$\pi_{0001000} = .0158653$	
$\pi_{1100010} = .0007743$	$\pi_{011100} = .0029449$	$\pi_{00111000} = .0056381$	$\pi_{00010000} = .0163192$	
$\pi_{110000} = .0007915$	$\pi_{0111000} = .0030708$	$\pi_{0011011} = .0057811$	$\pi_{0000111} = .0167865$	
$\pi_{1100000} = .0008125$	$\pi_{011011} = .0032024$	$\pi_{00110110} = .0059280$	$\pi_{00001110} = .0172710$	
$\pi_{101111} = .0008316$	$\pi_{0110110} = .0033435$	$\pi_{0011010} = .0060787$	$\pi_{0000110} = .0177688$	
$\pi_{1011110} = .0008525$	$\pi_{011010} = .0034904$	$\pi_{00110100} = .0062332$	$\pi_{00001100} = .0182838$	
$\pi_{101110} = .0008735$	$\pi_{0110100} = .0036430$	$\pi_{0011001} = .0063953$	$\pi_{0000101} = .0188159$	
$\pi_{1011100} = .0008964$	$\pi_{011001} = .0038051$	$\pi_{00110010} = .0065574$	$\pi_{00001010} = .0193634$	
$\pi_{101101} = .0009193$	$\pi_{0110010} = .0039749$	$\pi_{0011000} = .0067291$	$\pi_{0000100} = .0199298$	
$\pi_{1011010} = .0009441$	$\pi_{011000} = .0041561$	$\pi_{00110000} = .0069007$	$\pi_{00001000} = .0205135$	
$\pi_{101100} = .0009689$	$\pi_{0110000} = .0043430$	$\pi_{0010111} = .0070819$	$\pi_{0000011} = .0211181$	
$\pi_{1011000} = .0009956$	$\pi_{010111} = .0045394$	$\pi_{00101110} = .0072650$	$\pi_{00000110} = .0217418$	
$\pi_{101011} = .0010242$	$\pi_{0101110} = .0047492$	$\pi_{0010110} = .0074558$	$\pi_{0000010} = .0223865$	
$\pi_{1010110} = .0010528$	$\pi_{010110} = .0049686$	$\pi_{00101100} = .0076503$	$\pi_{00000100} = .0230484$	
$\pi_{101010} = .0010833$	$\pi_{0101100} = .0051975$	$\pi_{0010101} = .0078525$	$\pi_{0000001} = .0237388$	
$\pi_{1010100} = .0011157$	$\pi_{010101} = .0054397$	$\pi_{00101010} = .0080585$	$\pi_{00000010} = .0244464$	
$\pi_{101001} = .0011501$	$\pi_{0101010} = .0056972$	$\pi_{0010100} = .0082721$	$\pi_{0000000} = .0251789$	
$\pi_{1010010} = .0011825$	$\pi_{0101000} = .0062465$	$\pi_{00101000} = .0084915$	$\pi_{00000000} = .0259380$	
$\pi_{101000} = .0012207$	$\pi_{010011} = .0065460$	$\pi_{00100110} = .0089511$		
$\pi_{1010000} = .0012588$	$\pi_{010100} = .0059642$	$\pi_{0010010} = .0091896$		
$\pi_{100111} = .0012989$		$\pi_{0010011} = .0087165$		

15.5.2 Comparative Bandwidth

We set the general parameters as follows, according to the IEEE 802.11b norm. The SIFS and DIFS times are set to $20\mu s$ and $50\mu s$ respectively. The time slot interval for CRP is set to $20\mu s$. The size of the payload of a packet is set to 1,500 bytes. A packet (either regular or ACK) contains a physical heading of $96\mu s$. The MAC head and tail, are equal to 19 bytes in a regular packet and 14 bytes in an ACK one, are transmitted at the maximum speed, that is, 11 Mbit/s. Our simulator was written from scratch in C, and supposes perfect pairwise communication. We now further describe the specifics of each protocol.

- The 802.11b norm [1]. Each station has a *CW* parameter. At the beginning, a station chooses a κ – called backoff counter – in the interval $\{0,\dots,CW-1\}$. If $\kappa = 0$, the transmission begins immediately. Otherwise, if an empty time slot is observed, κ is decreased by one. At the end of a transmission, the value of *CW* itself is updated to *CWMin* if the transmission was successful, and to $\min(CWMax, 2*CW)$ if a collision occurred. We have set, as in the norm, $CWMin = 32$ and $CWMax = 1024$.
- The Idle Sense method [13]. At the end of a transmission, successful or not, the terminal stores the number of idle time slots before its transmission. After five transmissions, the terminal computes the average number of time slots waited for. If this number is inferior to 5.68, the congestion window is updated by

    $CW = \min(CWMax, CW * 1.2).$

    Otherwise, the new *CW* is given by

    $CW = \max(CWMin, 2*CW/(2+1e-3*CW)).$

- The additive congestion window increase/decrease [12]. At the end of an unsuccessful transmission, *CW* is set to $\min(CWmax, CW+32)$. If the transmission is successful, the station flips a biased coin, and with probability 0.1809 updates *CW* by

    $CW = \max(CWmin, CW-32),$

    and otherwise does not change *CW*.
- The CONTI method [3]. At each step a CRP of six time slots is applied with the probabilities given by Table 15.3. The surviving stations transmit.
- Our method – Selective tournaments. We apply a CRP of six to eight time slots. We have used the probabilities of Table 15.2. A station uses sequence w for signaling with probability π_w.

Our results are presented in Figure 15.6. In this figure, we plot for various numbers of stations the total throughput observed in the system. We clearly see that all the proposed methods improve significantly the original IEEE 802.11b mechanism. The methods based on adaptive tuning of the congestion window, namely [12, 13], achieve quite close performances. The CONTI method performs very well. Our method gives the best performance in all cases, and has a total improvement of as much as 34.1% for 100 stations over the original norm.
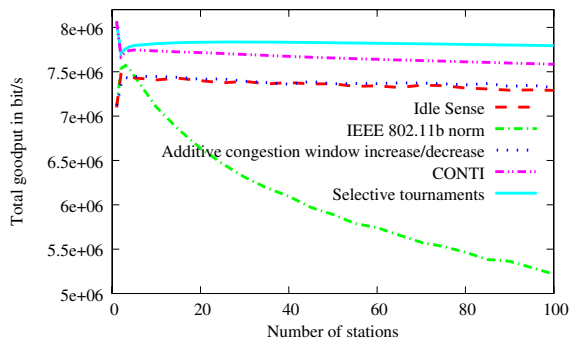
Fig. 15.6 Comparative total throughput for different protocols
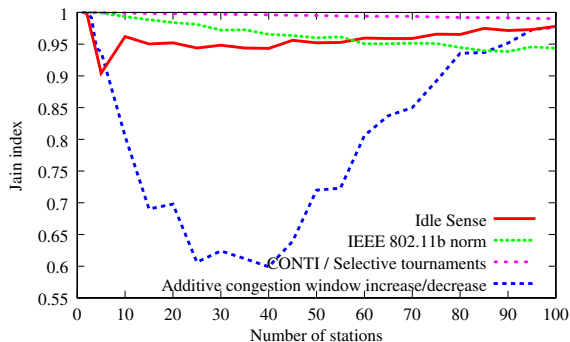
Fig. 15.7 Comparative Jain index for different protocols

15.5.3 Fairness Considerations

We undertake in this part more consideration of fairness issues. For each of the experiments, we have observed a series of 10,000 successful transmissions, and assigned to each station i the number x_i of packets it managed to transmit. In order to evaluate the fairness, we use the Jain index [15], defined by

$$Index = \frac{(\sum x_i)^2}{n \sum x_i^2}.$$

This index is always between 0 and 1, and closer to 1 if the system is more fair. Our results are given in Figure 15.7. Note that our method is equivalent to the CONTI method from the fairness point of view. The results plotted are averages obtained after a series of 10 tests.

Table 15.3 Values taken by CONTI

$l(w)$	0	1	2	3	4	5
p_w	0.07	0.2	0.25	0.33	0.4	0.5

The results show different behaviors. We observe as in [13] that slow conges-
tion window methods tend to generate some unfairness. We also notice that the new
method hardly improves the quality of the original IEEE 802.11b norm. Note, any-
way, that our method achieves the best fairness performance.

15.6 Conclusion

In this chapter we have demonstrated the efficiency of selective tournaments in the
wireless context. We have determined their limits in terms of avoidance of colli-
sion, and shown that they perform very well in terms of fairness. The tuning that we
propose achieves to our knowledge the best throughput performance in the 802.11b
framework. This advocates for a more extensive use of these methods, and the build-
ing of devices including this new access control mode. This is not necessarily a sim-
ple task, since the proposed scheme is not compatible with the previous ones, except
with CONTI, but is a promising way to achieve better wireless networks.

References

1. Higher-speed physical layer extension in the 2.4 GHz band. IEEE Std 802.11b-1999 Part 11:
   wireless LAN medium access control (MAC) and physical layer (PHY) specifications
2. IEEE standard for local and metropolitean network. Part 16: Air Interface for Broadband
   Wireless Access System (IEEE802.16 REVd D5-2004) (2004)
3. Abichar, Z., Chang, M.: CONTI: constant-time contention resolution for WLAN access. In:
   Proceedings of Networking, LNCS 3462, pp. 358–369 (2005)
4. Beuerman, S., Coyle, E.: The delay characteristics of CSMA/CD networks. IEEE Transactions
   on Communications **36**(5), 553–563 (1988)
5. Bianchi, G.: Performance analysis of the IEEE 802.11 distributed coordination function. IEEE
   Journal on Selected Areas in Communications **18**(8), 535–547 (2000)
6. Bianchi, G., Tinnirello, I.: Kalman Filter Estimation of the Number of Competing Terminals
   in an IEEE 802.11 network. In: IEEE INFOCOM, vol. 2, pp. 844–852 (2003)
7. Bononi, L., Conti, M., Gregory, E.: Optimization of IEEE 802.11 wireless LANs performance.
   IEEE Transactions of Parallel and Distributed Systems **15**(1), 66–80 (2004)
8. Cali, F., Conti, M., Gregori, E.: Dynamic tuning of the IEEE 802.11 protocol to achieve a
   theoretical throughput limit. IEEE/ACM Transactions on Networking **8**(6), 783–799 (2000)
9. Capetanakis, J.: Generalized TDMA: the multi-accessing tree protocol. IEEE Transactions on
   Communications **COM-27**(10), 1476–1484 (1979)
10. Capetanakis, J.: Tree algorithms for packet broadcast channels. IEEE Transactions on Infor-
    mation Theory **IT-25**(5), 505–515 (1979)
11. Fayolle, G., Flageolet, P., Hofri, M., Jacquet, P.: Analysis of a stack algorithm for random
    multiple-access communication. IEEE Transactions on Information Theory **IT-31**(2), 244–
    254 (1985)

12. Galtier, J.: Optimizing the IEEE 802.11b performance using slow congestion window decrease. In: Proccedings of the 16th ITC Specialist Seminar on performance evaluation of wireless and mobile systems, pp. 165–176. Antwerpen (2004)
13. Heuse, M., Rousseau, F., Guillier, R., Duda, A.: Idle sense: An optimal access method for high throughput and fairness in rate diverse wireless LANs. In: Proceedings of SIGCOMM 05. Philadelphia, USA (2005)
14. Ibrahim, M., Alouf, S.: Design and Analysis of an Adaptive Backoff Algorithm for IEEE 802.11 DCF mechanism. In: Proceedings of Networking 2006 Conference, *Lecture Notes in Computer Science*, vol. 3976, pp. 184–196. Coimbra, Portugal (2006)
15. Jain, R., Chiu, D. M., Hawe, W.: A quantitative measure of fairness and discrimination for ressource allocation in shared computer system. Tech. Rep. DEC-TR-301, Digital Equipment Corporation (1984)
16. Jamieson, K., Balakrishnan, H., Tay, Y.: Sift: A MAC protocol for event-driven wireless sensor networks. Tech. Rep. 894, MIT Laboratory for Computer Science (2003). URL `citeseer.ist.psu.edu/jamieson03sift.html`
17. Janssen, A., de Jong, M.: Analysis of contention tree algorithms. IEEE Transactions on Information Theory **46**(6), 2163–2172 (2000)
18. Kleinrock, L., Tobagi, F.: Packet switching in radio channels: Part I – carrier sense multiple access modes and their throughput-delay characteristics. IEEE Transactions on Communications **23**(12), 1400–1416 (1975)
19. Seo, J. B., Lee, H. W., Cho, C. H.: Performance of IEEE 802.16 random access protocol – steady state queuing analysis. In: Proceedings of Globecom, WLC17-2 (2006)
20. Standard, E. T.: HIgh PErformance Radio Local Area Network (HIPERLAN) Type 1; Functional Specification (1996)
21. Tsybakov, B., Mikhailov, A.: Free synchronous packet access in a broadcast channel with feedback. Prob. Inform. Trans. **14**, 259–280 (1978)
22. Wu, H., Peng, Y., Long, K., Cheng, S., Ma, J.: Performance of reliable transport protocol over IEEE 802.11 wireless LAN: analysis and enhancement. In: Proceedings of INFOCOM'02 (2002)
23. Xiao, Y.: Saturation performance metrics of the IEEE 802.11 MAC. In: Proc. of The IEEE Vehicular Technology Conference (IEEE VTC 2003 Fall), pp. 1453–1457. Orlando, Florida, USA (2003)

Chapter 16
Topology Control and Routing in Ad Hoc Networks

Lenka Carr-Motyckova, Alfredo Navarra, Tomas Johansson, and Walter Unger

Abstract Mobile nodes with the ability to communicate with radio signals may form an ad hoc network. In this chapter special problems arising for these ad hoc networks are considered. These include range control, the reduction of interferences, regulation of power consumption, and localization.

16.1 Introduction

Mobile nodes with the ability to communicate with radio signals may form an *ad hoc network*. This chapter starts with a brief discussion of important concepts with respect to topology control and routing in ad hoc networks.

A Mobile Ad hoc Network (MANET) consists of mobile platforms: routers, multiple hosts, and wireless communication devices which are free to move about arbitrarily. The system may operate in isolation, or may have gateways to and interface with a fixed network. MANETs have several characteristics: dynamic topologies, bandwidth-constrained links, variable capacity links, energy-constrained operation.

Lenka Carr-Motyckova · Tomas Johansson

Department of Computer Science and Electrical Engineering, Luleå University of Technology, SE-971 87 Luleå, Sweden, e-mail: `lenka,tomasjo@sm.luth.se`

Alfredo Navarra

Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Via Vanvitelli 1, 06123 Perugia, Italy, e-mail: `navarra@dmi.unipg.it`

Walter Unger

Lehrstuhl für Informatik 1, RWTH Aachen University, Ahornstraße 55, D-52056 Aachen, Germany, e-mail: `quax@cs.rwth-aachen.de`

The following is a list of desirable qualitative properties of MANET control algorithms: distributed operation, loop freedom, on-demand-based operations or proactive operations, unidirectional link support.

Sensors

In a MANET, there is no necessity to fix the position of a battery or a solar-powered wireless device. This advantage is utilized in many applications, i.e., in areas of missing or too expensive infrastructure, or in areas where the devices have to be placed randomly, or where the devices are in private environments, like offices or meeting places [11, 38]. The wireless devices (or sensors) form a so-called *ad hoc* wireless network. At start-up, ad hoc wireless networks do not have a known structure. There is just a collection of devices that may communicate with some of its neighbors using a common protocol. Consequently, several problems of self-organizing and economizing with the resources have to be considered for these networks.

Range Control and Localization

If the devices are equipped with *range control* technology, they could choose the power of the transmission signal to limit the engendered interference without losing too much of the transmission range. This prolongs also the lifetime of the batteries. Another problem is the unknown structure of the network at start-up. A protocol for routing the messages to the designated destinations has to be provided. One way to accomplish this is by *localization*; the nodes use some local information, e.g., the set of reachable neighbors, to compute a rough picture of the relative positions. This information could be used to solve further problems.

Topology Control

Routing is considered to be one of the most important problems in ad hoc wireless networks as it is influenced by specific features of these networks: limited battery power and communication through relaying by intermediate nodes. A transmission graph is defined by signal propagation and the interference caused by simultaneous transmissions to the same node. A transmitting edge disturbs all nodes in its vicinity. Each node is assumed to have a transmission radius that is needed to reach all its neighbors in a spanning tree. The interference must be minimized but at the same time the topology graph must be connected. Topology control algorithms are used to create subgraphs like spanners or low degree or sparse graphs. These topologies are known to have limited interference and power consumption. Creating such subgraphs is done by selecting a subset of the available links in the network graph $G = (V, E)$ to form a reduced graph $GTC = (V, ETC)$. The general approach of a

topology control algorithm is to remove long links from the network in order to force the nodes to use several short hops instead. A routing algorithm has to find a minimum power path: because the energy consumption grows quadratically with the distance, we need to use small hops. A short hop can be realized by relatively low transmission power. On the other hand, if a wrong selection of edges is removed, the paths become unacceptably long with respect to the number of hops, or the network may even become disconnected.

The routing objective might be to minimize the energy consumed for each message, to minimize the ratio of energy cost per packet, to minimize the maximum node energy cost (over the entire network), to maximize the lifetime of the network, to maximize the time to the earliest time a message cannot be sent, or to maximize the total number of messages successfully carried by the network.

Another energy saving strategy, *lazy scheduling*, allows the lowest possible transmit power for the longest possible time. Therefore, we observe a trade-off connectivity and sparseness: a sparse topology allows for some nodes to go to a sleep mode, but this harms the connectivity that is needed for small hop routing. Hence, transmission power of other nodes should be adjusted to topological changes. Usually ad hoc network models are based on unit disk graphs: nodes are connected by a link if and only if the Euclidean distance is at most 1. A trade off between scaling and spanning properties can be observed. Scaling requires that nodes have a constant number of neighbors. This implies that the resulting graph is sparse and nodes might be connected over a longer path than strictly necessary. A similar trade off works with nodes: too few nodes cause high energy cost, but too many nodes cause interference.

Clustering

Clustering constitutes a special kind of topology control. The purpose of clustering is to find a subset of nodes (cluster heads) such that the rest of the nodes are visible to at least one cluster head. Cluster heads are defined either as a maximum independent set [32] or a connected dominating set, or other criteria are applied. A virtual backbone built of cluster heads is responsible for routing. A routing graph consists of edges connecting clients with cluster heads and cluster heads with gateways. Data packets in the cluster graph are routed between gateways and cluster heads. The cluster structure of a network allows local or hierarchical routing that cuts down routing overhead and interference during transmissions.

Outline

In the following sections we consider problems that are typical for ad hoc networks. In Section 16.2 we consider the issue of reducing interference by means of topology control in an ad hoc network. Low interference prohibits retransmissions, which means a lower consumption of resources in intermediate nodes. Section 16.3 ex-

plains the energy-aware scatternet formation and routing in Bluetooth networks. Bluetooth networks require a special topology that is necessary for proper functioning of the network. In Section 16.4 we deal with the bandwidth-constrained routing. We focus on creating a cluster topology which organizes inter-cluster and intra-cluster communication separately. This kind of a hierarchical routing saves energy in the network, because nodes communicate mainly with the cluster-heads. Cluster-heads create a wireless backbone that accomplishes the inter-cluster communication. The hierarchical routing decreases the size of the routing tables, the number of routing updates, and therefore the energy needed for routing. The issue of the self-localization problem for each device is presented in Section 16.5.

16.2 Reducing Interference in Ad Hoc Networks

A collection of topology control algorithms that reduce interference in ad hoc networks will be presented in this section. Different metrics for interference will be discussed. Especially, the API algorithm (considering a routing issue) will be stressed here.

Transmitting nodes influence the ability of other nodes to receive data. A node is not able to receive data from its neighbor if another neighbor is transmitting at the same time. This mutual disturbance of communication is called interference. Reducing interference in the network leads to fewer collisions and packet retransmissions, which indirectly reduces power consumption and extends the lifetime of the network. Therefore, reducing the interference is an important goal for topology control and energy saving algorithms. This is achieved by selecting only a subset of the available links to be used for transmission to reduce the amount of interference.

An important question is how the amount of interference in a network should be measured. Let us recall that a transmission graph of a network is defined by signal propagation of every node in a network. A set of neighbors of node u is defined as a set of nodes that are able to receive a signal from u. The interference of the entire network is defined as the maximum edge coverage $Cov(e)$ in [4]: the maximum number of nodes affected by one specific link in the network. The authors show that there is no local algorithm to find the topology that gives the lowest maximum edge coverage, since knowledge of the entire network topology is needed. However, they present the algorithm LISE (Low Interference Spanner Establisher) that solves a similar problem: find the graph with the lowest possible maximum edge coverage that also is a t-spanner, where t is a constant that can be chosen freely. A t-spanner S for graph G is a subgraph such that distance $d_{S(p,q)} \leq t d_{G(p,q)}$, for any two vertices p, q of G.

This work is expanded upon in [28], where an alternative, receiver-centric, interference model is introduced. Unit disk graph UDG is used as a network graph: two nodes are connected with an edge if and only if they are at distance at most 1. In this model, the coverage of a node v in the UDG G is defined as the number of nodes

covering v with their disks induced by their transmission ranges. The interference of the entire network is defined by the maximum coverage for any node in the network.

In [24] an alternative interference metric that corresponds to the average interference of the entire network is presented. The interference is defined as the sum of the edge coverage of all edges in the network, divided by the number of nodes in the network n.

$$I(G) = \sum_{u,v \in V} Cov(e)/n$$

A different interference model is presented in [13]. This metric also takes the transmission power into account. If the number of neighbors is constant when a node increases or decreases its transmission power level from P_1 to P_2, the interference measure should increase or decrease as well. Also, if two nodes N_1 and N_2 are using the same transmission power level, but have different numbers of neighbors, the interference measures of the two nodes should differ to reflect the difference in the number of neighbors.

Other topology control algorithms are often based on computational geometry structures, such as the minimum spanning tree [27], or the Delaunay triangulation [12]. In [29], Rodoplu and Meng present an algorithm that keeps all energy-optimal paths. Their topology, which takes an energy model as input, is a general version of the Gabriel graph [8]. The XTC topology control algorithm [34] is shown to produce a subgraph of the original graph such that an edge between nodes u and v cannot exist if a node w exists such that $dist(uv) \geq max(dist(uw), dist(vw))$.

In this section we discuss the Average Path Interference (API) topology control algorithm, originally introduced in [16]. API is a topology control algorithm that minimizes the average path interference. The average path interference of a graph is defined as the sum of interference for all interference-optimal paths between node pairs, divided by the number of all node pairs in the graph. The interference-optimal path between nodes u and v is the path $IoptPuv = \{e_1, e_2, \ldots, e_k\}$ between u and v that has the lowest interference, according to the following definition: the interference of a path is defined as the sum of the coverage of all edges in the path, according to the definition of edge coverage in [4]. In other words $TotIoptPI(G)$ is a sum of interferences of interference-optimal paths between all node pairs in a graph.

$$TotIoptPI(G) = \sum_{u,v \in V} \sum_{e \in IoptPuv} Cov(e)$$

If the resulting value is divided by the total number of node pairs that are connected in G, we will get the average interference for all minimum-interference paths. A metric that considers the interference when shortest path routing is used can be obtained by replacing the interference-optimal path IoptPuv with the shortest path $SPuv$:

$$TotSPI(G) = \sum_{u,v \in V} \sum_{e \in SPuv} Cov(e)$$

In other words $TotSPI(G)$ is a sum of interferences of shortest paths (in the number of hops) between all node pairs in a graph. The average interference for all shortest paths is defined as $TotSPI(G)$ divided by the number of node pairs connected in G.

The maximum interference difference metric is defined as the largest difference in interference between $IoptPuv$ in the original graph and $IoptPuv$ in the graph that is the output of a topology control algorithm.

The Average Path Interference (API) algorithm [16] consists of two steps: First a Gabriel subgraph of the original graph is computed. A Gabriel Graph is defined as follows: there is an edge between u and v if there is no vertex w in the circle with diameter chord (u,v). The next step is to remove links that lead to high interference. The coverage of each link can be calculated locally. If a link can be replaced with two links that together have smaller coverage, it is removed. The graph produced by the algorithm is an energy spanner. For the proof we refer to [16]. The API algorithm was compared to XTC and LISE algorithms by simulations. The API topology gives results close to those of the XTC algorithm, and far better than LISE in the following measures:

Using the average interference-optimal path metric, the API algorithm generally gives the best results. Both the LISE and the API algorithms preserve almost all interference-optimal paths (cf. Figure 16.1). When considering the average path interference with respect to the shortest path, API and XTC give a lower interference than LISE (cf. Figure 16.2). Using the maximum interference difference metric, the API topology has lower interference than both LISE and, especially, XTC algorithms (cf. Figure 16.3).
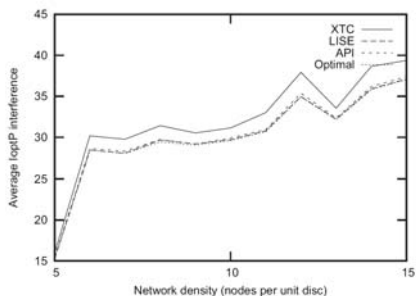
Fig. 16.1 The average interference-optimal path interference
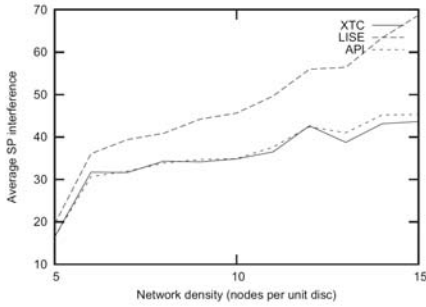
Fig. 16.2 The average shortest-path interference
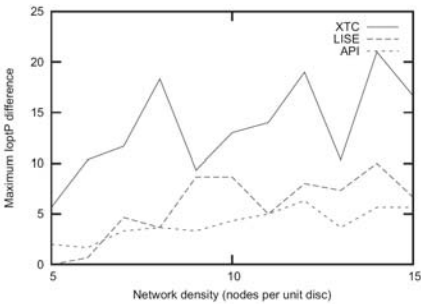
Fig. 16.3 The biggest difference between interference in a interference-optimal path compared to the interference-optimal path in the original topology

16.3 Energy Aware Scatternet Formation and Routing

In this section we consider energy-efficient routing in Bluetooth networks. These networks must obey certain rules imposed on their topology. Whenever two Bluetooth devices communicate, one must assume the role of a master and the other the role of a slave. Together they form a piconet. Each piconet consists of one master and an unlimited number of slaves, but there can only be at most seven active slaves simultaneously. If there are more than seven slaves in a piconet, some of them must therefore be parked, i.e., inactive. A collection of piconets is called a scatternet. In order to communicate between piconets, two nodes from neighboring piconets must get connected. A node can switch between master and slave mode on the time bases. In the same manner, a node can change a membership in different piconets at different time slots.

Recently, there has been an increased interest in routing algorithms in ad hoc networks that take the energy levels of the nodes into account. An example of an energy-aware routing algorithm is the Capacity-competitive (CMAX) algorithm [17], where each link in the network is assigned a weight. The weight of a link increases with the energy consumed by the sending of a unit message over the link,

and with the increase of energy utilization of the node when it transmits message m over the link. The shortest path with respect to link weights is selected by the CMAX algorithm. This algorithm requires that each node knows the energy utilization of the other nodes in the network. In order to spread the power information through the network, each node periodically broadcasts its power information to the nearest nodes.

The max-min zP_{min} algorithm [20] finds paths that avoid nodes with only a small fraction of their battery power remaining. The goal is to find the path with the maximal minimal fraction of remaining power after the message is transmitted. However, the path must also not use more power than a constant factor of the smallest possible power consumption P_{min}. This results in finding a path that avoids using up the power of individual nodes, while still maintaining an upper bound on the total power consumption of the path.

Both of the above algorithms require that each node have full knowledge of the topology of the network, an assumption that is not realistic in larger networks.

Another approach is to extend existing *on-demand algorithms* in order to compute energy-efficient paths. On-demand routing algorithms create a route only after a demand for the routing of a message. The authors in [25] enhance the Ad hoc On Demand Distance Vector (AODV) routing protocol by adding energy information in the route request messages. In this way, the destination node can decide on the path with the lowest energy cost. The extended AODV algorithm only considers total energy cost and estimated bit error rates in order to select the path; it does not consider the amount of energy that remains in the nodes.

Wang et al. [33] present a power-aware on-demand routing protocol that takes the remaining energy of the nodes into account. The routing algorithm anticipates energy requirements of a transmission and makes energy reservation for it. The metric used for a path selection takes to account the shortest path and the maximum lifetime of the network.

Routing in Bluetooth networks presents more challenges than routing in general ad hoc networks. For every communication between two Bluetooth devices they must form a master-slave relationship. A Bluetooth node can only take part in one such relationship at a time. Therefore, creating a path between two Bluetooth nodes is not a trivial problem, especially if there are several simultaneous communications in the network that intersect each other. A proactive algorithm that creates a routing path in scatternet formation before there is a demand for transmission is described in [37]. A single node initiates the construction of a Bluetree. The root node assumes the role of master and selects all its neighbors as slaves, and the slaves in turn assign themselves as masters in new piconets and search for unassigned neighbors to be their slaves. This procedure is repeated until the entire network is covered by the Bluetree.

In [21] the authors use location information in order to produce a planar subgraph, which can improve the performance of routing algorithms. This optional step is followed by assigning roles to the Bluetooth nodes.

Recently, there has been work that aims to extend the lifetime of scatternets by considering the amount of power remaining for each node [36]: when a node wants

to initiate a communication, it sends out a route request to the destination by flood-
ing the network. When the destination receives the request it responds with a route
reply. To form the temporary scatternet between the source and the destination, the
destination sets its role to master, and for the rest of the nodes in the path every
other node is a master while the other nodes are slaves in two different piconets.
No node needs to know about the topology of the entire network. This approach
has a relatively simple mechanism to avoid low-powered nodes: either a node has
enough energy to forward a message or it does not. When a node is reached by a
route request there is no way to know from which path the packet comes and what
the energy condition of the entire path is.

In the following paragraph we consider routing data between scatternets in a
Bluetooth network over a path that consumes less energy compared to other possi-
ble paths [15]. The two main differences between the algorithm presented in [15]
and Wang's algorithm is that the latter does not take the total power cost of paths
into account, and the cost metric does not change over time to reflect the changing
amount of energy in the nodes. In [15] a creation of a routing path over a piconet
sequence is considered. The on-demand scatternet formation and routing algorithm
create a local, temporary scatternet that forms a path between the sending and re-
ceiving nodes for the duration of the data transfer. The scatternet is created just for
the purpose of a single transmission between the sender and the receiver, and the
existence of the scatternet (a sequence of piconets) is limited just to the time of data
transmission. It means that the topology (definition of piconets) is disregarded after
the transmission is completed.

The routing algorithm starts by flooding the network with a request for a route
(route request procedure). Several alternative paths between the source and the des-
tination are found, while measuring the energy parameters for each path. The idea
is to choose a path that avoids nodes that have a small amount of energy left, while
still keeping the total energy cost of the path as low as possible. When evaluating
the quality of a path (with respect to the energy levels in the nodes), the following
definitions will be used:

- the potential remaining power of every link (i, j) in the path as

$$rp_{ij} = p_i - e(m)_{ij}$$

  where p_i is the power level at node i, and $e(m)_{ij}$ is the power needed to send the
  message m directly from i to j.
- the amount of energy needed to send the message m over the path P: the metric
  that represents the energy consumption of the path P is defined as

$$Pow(P) = \sum_{i=1}^{n} e(m)_{l_i}$$

  where n is a sequence number of the last node in the path and l_i is the ith link in
  the path.

The minimum $Min(P) = \min_{l \in P} rp_l$ value for each path received during the route request procedure S is computed as the minimum potential remaining power of all links in path P. The threshold value $rp_{threshold}$ is defined as a median of $Min(P)$ over all received paths in S. The basic idea is to avoid nodes which would be drained of power if they were to forward the data message. In order to achieve that, all paths $P \in S$ such that

$$Min(P) < rp_{threshold}$$

are canceled. Of the remaining paths in S, the path P with the lowest $Pow(P)$ is chosen.

In order to evaluate the algorithm, simulations were performed to compare it with a routing algorithm that always selects the path that consumes the least amount of energy. Each node started with the same amount of energy. At every time step, data transmissions between randomly selected sender and receiver nodes were generated. This was repeated until the first node in the network had depleted all its power. The results from the simulations showed that the algorithm in [15] extends the lifetime of the network compared to choosing energy-optimal paths without taking the power level of the nodes into account. Figure 16.4 show the average distribution of energy in the network when the first node had depleted its power. The presented algorithm, represented by the dashed line in the graph, had consumed more energy (at least partly due to the fact that it had routed more messages), but the load had been spread somewhat more evenly through the network.
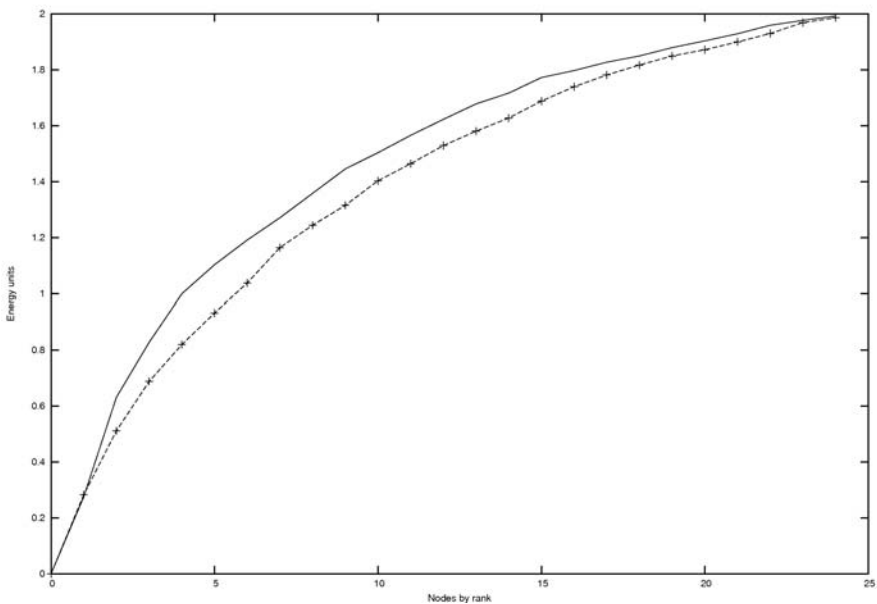
Fig. 16.4 Remaining energy in nodes

16.4 Bandwidth-Constrained Clustering

The purpose of clustering algorithms is to divide the original network graph into non-overlapping subgraphs. The resulting structure is used for different control functions, including routing. The control functions decrease interference and the amount of global traffic, and therefore decrease energy consumption of the network.

In this paragraph we consider only clusters, where the *slaves* are direct neighbors of its *cluster head*. Recent work in clustering for wireless networks began with the work of Gerla and Tzu-Chieh Tsai [10]. In their algorithms, if a node hears from a cluster head with a lower ID than itself, it resigns and uses that node as a cluster head instead. Another version uses the degree of the nodes. The idea is that nodes with a high degree are good candidates for cluster heads, since the resulting clusters will be larger. However, even small changes in the network topology can result in large changes in the degree of the nodes. This means that the cluster heads are not likely to stay as cluster heads for a long time, and the clustering structure becomes unstable. On the other hand, using the Lowest-ID algorithm, the nodes with a low ID stay as cluster heads most of the time. This results in an unfair distribution of load that could lead to some nodes losing power prematurely. Amis and Prakash [1] present additions to these clustering mechanisms that help avoid cluster head exhaustion by providing *virtual IDs* to the nodes.

An algorithm that makes it possible to choose clusters with a radius r larger than 1 (the slaves are r hops away from its head) is presented in [2]. This algorithm produces large clusters that are relatively stable compared to the previously mentioned algorithms. This algorithm also uses the node's ID values when forming the clusters. First, the nodes set their *winner value* (possible leader ID) to be their ID number, and broadcast it. If one node receives a larger winner value than its own, it switches to the new winner value instead. This procedure is repeated r times. The result is that the larger ID values spread through the network. The process is repeated, except that lower winner values now overtake larger ones. The purpose is to achieve a balance in the cluster sizes, instead of having the clusters with the largest IDs be much larger than the others. If clusters of constant size is the primary objective, this algorithm is the only one that guarantees the property. In [9], another algorithm creates a cluster structure that is, with high probability, a constant approximation of the optimal solution. In this case, an optimal cluster structure is the one that uses the lowest number of clusters of radius r to cover the network at this time.

McDonald and Znati [23] present an algorithm that forms clusters of nodes that have sufficient probability to stay connected during a specific time interval. The algorithm requires that movements of nodes in an ad hoc network are predictable, something which might not always be true. In [22], Lin and Gerla present an algorithm that dynamically maintains clusters in a dynamic environment. A cluster-based energy conservation algorithm including the cluster formation is described in Xu et al. [35]. Ryu, Song, and Cho [30] suggest that by using a distributed heuristic clustering scheme, the transmission power can be minimized. A similar problem of power control supported by clustering is solved in Kawadia and Kumar [18]. A

hierarchical clustering proposed by Bandyopadhyay and Coyle [3] is used to save energy in wireless sensor networks.

Most existing clustering algorithms create new clustering structures from scratch after a specified time interval in order to maintain cluster structure properties. In [14] the maintenance function is interleaved with the traditional clustering function. The algorithm consists of two parts, the clustering part, where a clustering structure is created from scratch, and the maintenance part, where the existing clustering structure is modified where necessary.

The clustering part of the algorithm starts with a broadcast phase. Nodes broadcast their leader values (initialized to the node's ID) to all the neighbors, and wait for broadcasts from all of them. When a node receives a value that is higher than its own, it sets its leader value to the received value. When all nodes have exchanged their messages, one round of the broadcast phase is completed. There are r broadcast phases, where r, a parameter of the algorithm, is the maximum radius of the clusters that are created. At the end of the last broadcast phase, the larger ID values have spread through the network.

Once the clustering part is completed, the *maintenance* part of the algorithm is performed. If the path to the cluster leader, which is checked by regular messages, does not exist anymore, the node will try to find a new path to a cluster leader through one of its reachable neighbors. Otherwise, it becomes an *orphan* node and starts up the clustering part of the algorithm.

The algorithm by Johansson and Carr [14] has time complexity of $\mathcal{O}(r)$, where no node is more than r hops away from the cluster head. Since r is likely to be a very small constant, this is an acceptable complexity. The overall message complexity is low, due to the nature of the algorithm and the maintenance part of the algorithm.

The number of messages depends largely on the radius of the clusters created. While it might be advantageous to create clusters with radius larger than 1, it is important to avoid unnecessary broadcasting. The algorithm only uses one broadcast phase. Clusters created by the algorithm have limited radius. The properties of different clustering algorithms covered here are summarized in Table 16.1.

16.5 Localizing Using Arrival Times

In this section, the problem of localizing random sensor networks is considered. There are two approaches: the first one uses information about the reachable neighbors and the second one uses the arrival times of messages within the network.

We shortly explain the first approach. If the sensors of a network are laid out on the plane, then some of the nodes form the outer boundary of the network. There may be more boundaries within the network, due to some holes. These boundary nodes may be identified, because they have a lower degree than the inner nodes. In a second step each node may compute the minimal number of hops to any border node. The nodes where the distance to the border is maximal compared to the local neighbors form the backbone of the structure of the network. Now disjoint groups

Table 16.1 Summary of different clustering algorithms

Algorithm	Properties	Complexity	Strengths	Weaknesses
Lowest-ID (LCA2) [10]	Cluster head selection based on node ID. Cluster head is directly linked to any other node in the cluster.	Constant time complexity, message complexity increase with denseness of graphs.	Fast and simple algorithm. Relatively stable clusters.	Small clusters. Some cluster heads likely to remain for long time.
Highest-connectivity [10]	Cluster head selection based on highest degree, otherwise same as LCA2.	Same as LCA2.	The nodes with highest degree are good candidates for cluster heads.	Very unstable clusters.
Max-min d-cluster [2]	Cluster radius d, where d is a constant.	$\mathcal{O}(d)$ time and storage complexity.	Large and stable clusters.	High number of messages sent.
Discrete mobile centers. [9]	One-radius clusters are produced. The number of clusters is a constant-factor approximation of the smallest possible number.	$\mathcal{O}(sn)$ storage complexity, where s is usually small, but can be up to n. Time complexity $\mathcal{O}(\log\log n)$.	Close to optimal clustering structure with respect to number of clusters.	No simulations to show cluster stability of the algorithm.
Hierarchical [31]	No fixed diameter of each cluster. Cluster size $< k, 2k-1 >$, where k is a constant.	Time complexity $\mathcal{O}(E)$.	Guaranteed upper and lower bound on cluster size.	Slow algorithm. Cluster radius can be up to k.
Adaptive clusters [23]	Created clusters should be connected in time t with probability α.	Undefined.	α and t can be varied in order to adapt to different mobility rates.	Difficult to predict future connectivity.
Maintenance	Added maintenance phase.	$\mathcal{O}(r)$ where r is the radius of the clusters.	Repairs lost connections.	Small clusters.

of nodes may be formed using some of the backbone nodes. There are two reasons to form these disjoint groups. The first is that each node has precisely one leader. Secondly, it is now a more simple task to get the overall topology of the network. The topology of the groups provides enough information to localize the nodes of the network. This nice technique is presented in a series of papers [5–7, 19].

A second approach is presented in [26]. Random instances of sensor networks are studied inside a square area. The power of transmission P_s is fixed for each sensor. A small percentage of the sensors called *Anchors* are assumed to be equipped with GPS capabilities. Hence, those sensors have the knowledge of their actual position; all other ones are assigned a random estimation of their position. Moreover,

sensors are equipped with *Time of Arrival* (ToA) capabilities, i.e., they are able to estimate their distances to the corresponding neighbors. The network is asynchronous. Hence, sensors can be in one of two different operational states: *sleep* and *wake*. In the sleep state a sensor can receive the position communications from other sensors, and computes its new position accordingly. In the wake state a sensor communicates the information concerning its estimated position to its neighborhood. Each sensor is assumed to operate for a predetermined time interval.

anchors   masses   springs   the composed system
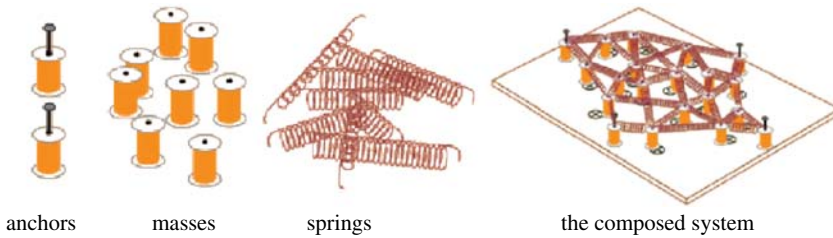
Fig. 16.5 Mass-spring system. Circles on the board of the composed system represent the real position of the corresponding masses

The modeling can be seen like a mass-spring system; see Figure 16.5. Sensors are masses. Masses representing anchors are well positioned in the area, while all others have a random estimation of their actual position. When a sensor performs a transmission, receivers can derive the distance at which the sender should be. This is accomplished by connecting senders and receivers by means of springs. The resting length of the spring is the length estimated by the ToA equipment, while the length assigned is equivalent to the distance of the estimated positions of the masses. Due to this, masses are subject to a set of forces generated by the springs. These forces tend to move the whole system to a final configuration of equilibrium. This is accomplished by means of successive transmissions from each sensor of its estimated position until the desired equilibrium is reached. That is, forces acting on the masses are smaller than a fixed threshold. As in a real mass-spring system, the algorithm makes use of two main parameters, i.e., the *damper* and the *elastic* constants of the springs. The former reflects the stability of springs with respect to oscillations. The latter reflects the ability of springs to recover and return to their original shape after being stressed or deformed. Those two parameters must be well tuned in order to obtain good performances of the convergence process to the equilibrium.

The *Localization Algorithm* exploits a framework that computes the dynamics of mass-spring systems. The algorithm, from now on called *Basic Localization* algorithm (*BL*), is composed of two phases: an *Initialization phase* and an *Propagation phase*.

In particular, in the *Initialization phase* a random distribution of sensors and anchors is simulated. Each sensor s_i computes the set N_{s_i} of sensors within its transmission range. Then, for each $s_j \in N_{s_i}$, a spring with endpoints s_i and s_j is created.

Initially, every anchor a_i communicates its position to each sensor $s_j \in N_{a_i}$ in order to give a first rough estimation of the sensors' positions. Then, each informed sensor communicates its estimated position to its neighbors that have not estimated their position yet. The Initialization phase ends when all sensors have an estimation of their positions. The obtained configuration will be referred as the *Initial Configuration*.

The *Propagation phase* computes the dynamics of the mass-spring system. Each mass s_i is subject to an internal force F_{s_i} that is the result of the forces generated by each spring connected to s_i. At each time step, each mass s_i modifies its position according to the internal force F_{s_i}. At the end of the Propagation phase, the final configuration of the mass-spring system approximates the *Target Configuration*, where the F_{s_i} acting on each mass s_i is close to zero. The *Propagation phase* ends when the force acting on each mass is less than a given threshold F_{toll}.

The *BL* algorithm can be modified in order to minimize the number of sensor transmissions needed to compute the *Target Configuration*. The gain with respect to the time required by *BL* to converge is even more considerable when a $\pm 1\%$ error of the sensors' ToA equipment is considered.

The following sensor localization strategies were proposed in order to improve the *BL* algorithm:

- AAD, *Ad Hoc Anchor Deployment strategy*: a limited number λ of anchors are deployed on the border of the square area.
- DES, *Dynamic Elastic constant Strengthening strategy*: springs connecting at least one anchor have the elastic constants increased by a factor $\gamma > 1$. Therefore, anchors have more influence in the localization problem computation.
- VA, *Virtual Anchors strategy*: if a sensor does not change its position for a given successive number of steps T, its status is moved to anchor.
- CA, *Computed Anchors strategy*: if a sensor has some fixed number K of anchors in its neighborhood, it computes its position from the positions of those anchors and becomes an anchor itself.

In [26], comparisons among strategies and combinations of such strategies have been evaluated for hundreds of instances under different assumptions with respect to the density of the network and the percentage of anchors.

16.6 Conclusions

In this chapter we have discussed selected algorithmic topics in the area of mobile ad hoc networking. In total, four topics were discussed.

In contrast to most of the related work, we studied in Section 16.2 at the interference of entire paths instead of interference of individual edges or nodes. Three new interference metrics that aim to reflect the interference of the entire network have been presented. A new topology control algorithm that produces an energy-spanning graph is also presented.

Clustering presents one of the approaches to decrease power consumption in ad hoc networks. In Section 16.4 an overview of different approaches to clustering in ad hoc networks is presented. We review a new bandwidth-constrained clustering algorithm that minimizes communication overhead, while still producing relatively large and stable clusters.

In Section 16.3 we presented the case of energy-aware scatternet formation that is used for routing in Bluetooth networks. The algorithm requires more traffic overhead compared to algorithms that do not take the power consumption into account. On the other hand, the algorithm distributes the data traffic as evenly as possible among all nodes in the network.

In Section 16.5 several localization algorithms are presented. For all these sections, good strategies and algorithms are presented. The consideration of the combination of the above problems remains open; also, a nice theoretical background is missing.

References

1. Amis, A. D., Prakash, R.: Load-balancing clusters in wireless ad hoc networks. In: Proceedings 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, pp. 25–32 (2000)
2. Amis, A. D., Prakash, R., P, T. H., Dung, V., Huynh, T.: Max-min d-cluster formation in wireless ad hoc networks. In: Proceedings of IEEE INFOCOM, pp. 32–41 (2000)
3. Bandyopadhyay, S., Coyle, E.: An energy efficient hierarchical clustering algorithm for wireless sensor networks. In: Proceedings of IEEE INFOCOM, vol. 3, pp. 1713–1723 (2003)
4. Burkhart, M., von Rickenbach, P., Wattenhofer, R., Zollinger, A.: Does topology control reduce interference? In: MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, pp. 9–19. ACM, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/989459.989462
5. Buschmann, C., Pfisterer, D., Fischer, S.: Estimating distances using neighborhood intersection. In: Proceedings of 11th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, pp. 314–321 (2006)
6. C.Buschmann, H.Hellbrück, S.Fischer, A.Kröller, Fekete, S.: Radio propagation-aware distance estimation based on neighborhood comparison. In: European Workshop on Sensor Networks, *Lecture Notes in Computer Science*, vol. 4373, pp. 325–340 (2007)
7. Fekete, S., Kröller, A., Buschmann, C., Fischer, S.: Geometric distance estimation for sensor networks and unit disk graphs. In: J. Gudmundsson, R. Klein, G. Narasimhan, M. Smid, A. Wolff (eds.) Geometric Networks and Metric Space Embeddings, no. 06481 in Dagstuhl Seminar Proceedings (2007)
8. Gabriel, K. R., Sokal, R. R.: A new statistical approach to geographic variation analysis. Systematic Zoology **18**, 259–278 (1969)
9. Gao, J., Guibas, L. J., Hershberger, J., Zhang, L., Zhu, A.: Discrete mobile centers. In: Discrete and Computational Geometry, pp. 188–196. ACM Press (2001)
10. Gerla, M., chieh Tsai, J. T.: Multicluster, mobile, multimedia radio network. Journal of Wireless Networks **1**, 255–265 (1995)
11. Hac, A.: Wireless sensor network designs. John Wiley & Sons, Ltd (2003)
12. Hu, L.: Topology control for multihop packet radio networks. IEEE Trans. on Communications **41**(10), 1474–1481 (1993)
13. Iannone, L., Khalili, R., Salamatian, K., Fdida, S.: Cross-layer routing in wireless mesh networks. In: Proc. ISWCS, pp. 319–323 (2004)

14. Johansson, T., Carr-Motyckova, L.: Bandwidth-constrained clustering in ad hoc networks. In: Proceedings of the Third Annual Mediterranean Ad Hoc Networking Workshop, pp. 379–385 (2004)

15. Johansson, T., Carr-Motyckova, L.: Energy-aware on-demand scatternet formation and routing. In: Proceedings of the 3rd International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks: LOCAN 2007 (2007)

16. Johansson, T., Carr-Motyčková, L.: Reducing interference in ad hoc networks through topology control. In: DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing, pp. 17–23. ACM, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1080810.1080815

17. Kar, K., Kodialam, M., Lakshman, T. V., Tassiuals, L.: Routing for network capacity maximization in energy constrained ad hoc networks. In: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, vol. 1, pp. 673–681 (2003)

18. Kawadia, V., Kumar, P. R.: Power control and clustering in ad hoc networks. In: Proc. IEEE Infocom, pp. 459–469 (2003)

19. Kröller, A.: Algorithms for topology-aware sensor networks. Ph.D. thesis, TU Braunschweig (2008)

20. Li, Q., Aslam, J., Rus, D.: Online power-aware routing in wireless ad-hoc networks. In: In MOBICOM, pp. 97–107 (2001)

21. yang Li, X., Stojmenovic, I., Wang, Y.: Partial delaunay triangulation and degree limited localized bluetooth scatternet formation. In: in IEEE Transactions on Parallel and Distributed Systems, pp. 17–32 (2003)

22. Lin, C. R., Gerla, M.: Adaptive clustering for mobile wireless networks. IEEE Journal on Selected Areas in Communications **15**, 1265–1275 (1997)

23. Mcdonald, A. B., Znati, T.: A mobility based framework for adaptive clustering in wireless ad-hoc networks. IEEE Journal on Selected Areas in Communications **17**, 1466–1487 (1999)

24. Moaveni-Nejad, K., Li, X. Y.: Low-interference topology control for wireless ad hoc networks. In: ACM Wireless Networks, pp. 41–64. IEEE Press (2005)

25. Nadeem, T., Banerjee, S., Misra, A., Agrawala, A.: Energy-efficient reliable paths for on-demand routing protocols. In: Sixth IFIP IEEE International Conference on Mobile and Wireless Communication Networks (2004)

26. Navarra, A., Tofani, A.: Distributed localization strategies for sensor networks. In: 4th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS), pp. 1–3 (2007)

27. Ramanathan, R., Rosales-Hain, R.: Topology control of multihop wireless networks using transmit power adjustment. In: INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 404–413. IEEE (2000)

28. von Rickenbach, P., Schmid, S., Wattenhofer, R., Zollinger, A.: A robust interference model for wireless ad-hoc networks. In: IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 12, p. 239.1. IEEE Computer Society, Washington, DC, USA (2005). DOI http://dx.doi.org/10.1109/IPDPS.2005.65

29. Rodoplu, V., Meng, T. H.: Minimum energy mobile wireless networks. IEEE Journal on Selected Areas in Communications **17**, 1333–1344 (1999)

30. Ryu, J., S. H.Song, Cho, D.: New clustering schemes for energy conservation in two-tiered mobile ad hoc networks. In: IEEE transactions on vehicular technology, vol. 51, pp. 1661–1668 (2002)

31. S.Banerjee, Khuller, S.: A clustering scheme for hierarchical control in multi-hop wireless networks. Tech. Rep. CS-TR 4103, University of Maryland, College Park (2000)

32. Schneider, J., Wattenhofer, R.: A log-star distributed maximal independent set algorithm for growth-bounded graphs. In: PODC '08: Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing, pp. 35–44. ACM, New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1400751.1400758

33. Wang, K., long Xu, Y., liang Chen, G., feng Wu, Y.: Power-aware on-demand routing protocol for MANET. In: ICDCSW '04: Proceedings of the 24th International Conference on

Distributed Computing Systems Workshops - W7: EC (ICDCSW'04), pp. 723–728. IEEE Computer Society, Washington, DC, USA (2004)

34. Wattenhofer, R., Zollinger, A.: XTC: A practical topology control algorithm for ad-hoc networks. In: Proc. of the 4 th Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN (2004)

35. Xu, Y., Bien, S., Mori, Y., Heidemann, J., Estrin, D.: Topology control protocols to conserve energy in wireless ad hoc networks. Tech. rep., Center for Embedded Networked Computing Technical Report 6 (2003)

36. Y.Liu, M. J.Lee, T. N.Saadawi: A bluetooth scatternet route structure for multihop ad hoc networks. IEEE Journal on Selected Areas in Communications **21**, 229–239 (2003)

37. Zaruba, G. V., Basagni, S., Chlamtac, I.: Bluetrees-scatternet formation to enable bluetooth based ad hoc networks. Communications, 2001. ICC 2001. IEEE International Conference on **1**, 273–277 (2001)

38. Zhao, F., Guibas, L.: Wireless sensor networks: an information processing approach. Morgan Kaufmann (2004)

Index